大数据时代的
数据分析利器!

信息科学与技术丛书

主　编　杨池然
副主编　张延召　侯延明　缪　敏

SAS开发经典案例解析

- SAS 编程语法基础
- 数据步基础与案例
- 过程步基础与案例
- 函数基础与案例
- 宏基础与案例
- Oracle 交互应用
- SAS 与关系数据库
- 统计分析基础与案例
- ODS 基础与综合案例
- 信用卡管理系统案例
- SAS 模型开发案例


网上提供源代码下载
www.cmpbook.com




机械工业出版社
CHINA MACHINE PRESS

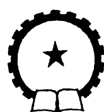
VISIT…

LANZAROTE Caliente.com

信息科学与技术丛书

SAS 开发经典案例解析

主　编　杨池然
副主编　张延召　侯延明　缪　敏
参　编　阮小东　郁鑫洲　孙焕之　赵霁红
　　　　陈　平　陈剑平　邢武当



机械工业出版社

本书以经典案例的形式讲解SAS的实际应用。全书共分12章，主要内容包括SAS概述与整体架构、SAS编程语法基础、数据步基础与案例、过程步基础与案例、函数基础与案例、宏基础与案例、统计分析基础与案例、SAS与关系数据库Oracle交互应用、ODS基础与综合案例、信用卡管理系统案例、SAS模型开发案例和高级应用技巧。

本书可以作为SAS爱好者、本科生学习SAS语言实践应用的教材，也可作为各类SAS工作者的参考书。

图书在版编目（CIP）数据

SAS开发经典案例解析/杨池然等编著. —北京：机械工业出版社，2013.1（2014.4重印）

（信息科学与技术丛书）

ISBN 978-7-111-41100-0

Ⅰ. ①S… Ⅱ. ①杨… Ⅲ. ①统计分析－应用软件 Ⅳ. ①C819

中国版本图书馆CIP数据核字（2013）第007499号

机械工业出版社（北京市百万庄大街22号 邮政编码100037）

策划编辑：丁 诚

责任编辑：丁 诚 范成欣

责任印制：乔 宇

保定市中画美凯印刷有限公司印刷

2014年 4月•第1版第 2次印刷

184mm×260mm•22.75印张•563千字

3001－4020册

标准书号：ISBN 978-7-111-41100-0

定价：59.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务	网络服务
社服务中心：（010）88361066	教 材 网：http://www.cmpedu.com
销 售 一 部：（010）68326294	机工官网：http://www.cmpbook.com
销 售 二 部：（010）88379649	机工官博：http://weibo.com/cmp1952
读者购书热线：（010）88379203	**封面无防伪标均为盗版**

前 言

随着“大数据”时代的到来，国内对数据仓库中的海量数据的ETL处理、海量数据的分析、数据挖掘以及数据模型的建设人才的需求逐年增多。SAS软件经过30多年的发展已经成为一款优秀的国际标准海量数据统计分析、数据挖掘、模型创建、海量数据处理开发软件，在国内得到了快速的发展和广泛的应用。为满足各类学习者的实际应用需求，本书作者从SAS数据编程实践案例、统计分析应用案例、SAS与关系数据仓库Oracle结合的金融应用案例、信用卡管理系统案例、模型开发案例以及高级技巧案例等方面深入剖析讲解SAS的实际应用案例。

全书共分3部分。第1部分是基础篇，包括第1章和第2章。通过基础篇的实际应用案例的讲解可以让学习者实际运行每一个案例程序，而不是停留在理论的讲解上，从而对 SAS系统有一个整体的认识。

第2部分是提高篇，包括第3～9章。通过数据步对大数据量的处理的实际应用案例详细讲解SAS数据步应用的方法和技巧。通过对过程步实际应用案例综合应用的讲解让读者理解过程步与数据步结合应用的实际价值。通过对函数与宏过程实际案例的讲解帮助读者理解并掌握SAS函数与数据步结合应用以及与宏过程结合应用的实际应用方法和技巧。通过对宏过程实际案例的讲解帮助读者理解宏功能的封装。统计分析实际案例解读是SAS在统计分析领域的应用讲解。SAS与关系数据库Oracle交互应用实际案例体现了SAS数据交互处理的能力，提取数据库中的数据，从而实现对海量数据的数据分析与数据挖掘。ODS输出实际案例对SAS数据分析结果输出到各类文件进行了详细的讲解和语法解读。

第3部分是综合实战篇，包括第10～12章。第10章通过对信息卡管理系统案例的详细讲解，帮助读者快速掌握 SAS 实际应用开发的步骤、流程、技巧以及应注意的问题。第 11章通过对SAS模型开发案例的详细讲解，帮助读者理解整个模型创建的流程、开发技巧以及注意事项。第 12 章通过对高级应用技巧案例的详细讲解，帮助读者写出优秀的、高效率的SAS程序。

本书由杨池然主编，参加本书编写的人员还有张延召、侯延明、缪敏、阮小东、郁鑫洲、孙焕之、赵霁红、陈平、陈剑平和邢武当，在此表示感谢，同时也感谢同事、家人和朋友的支持。由于作者水平有限，加上时间仓促，书中难免存在不妥与疏漏之处，欢迎广大读者批评指正，以便于以后修订。

编 者

目　　录

前言

第 1 篇　基　础　篇

第 1 章　SAS 概述与整体架构 …… 1
1.1　SAS 概述 …… 1
1.1.1　SAS 介绍 …… 1
1.1.2　SAS 行业应用现状 …… 1
1.2　SAS 整体架构 …… 2
1.2.1　SAS 整体架构流程 …… 2
1.2.2　SAS 各模块实现功能 …… 4
第 2 章　SAS 编程语法基础 …… 6
2.1　逻辑库定义与应用案例 …… 6
2.1.1　逻辑库的作用与定义 …… 6
2.1.2　Windows 环境与 UNIX 环境创建逻辑库 …… 8
2.1.3　逻辑库与关系数据库的连接方式 …… 8
2.2　SAS 编程语法 …… 9
2.2.1　变量与常量 …… 10
2.2.2　条件选择语句与循环语句 …… 13
2.2.3　操作符与宏变量 …… 18
2.2.4　格式修饰符与指针控制 …… 21
2.2.5　INPUT 语句与 PUT 语句 …… 26
2.2.6　INFILE 语句与 FILE 语句 …… 30
2.2.7　DELIMITER 语句与 LENGTH 语句 …… 35
2.2.8　日期与时间定义 …… 37
2.2.9　INFORMAT 与 FORMAT 定义数据格式 …… 38
2.2.10　LABEL 语句与 RETAIN 语句 …… 39
2.2.11　RENAME 语句与数组语句 …… 40
2.2.12　SAS 编程注释与 OPTIONS 语句 …… 41

第 2 篇　提　高　篇

第 3 章　数据步基础与案例 …… 44
3.1　数据步基础 …… 44
3.1.1　数据步概述与定义 …… 44
3.1.2　SET 语句 …… 49
3.1.3　MERGE 语句 …… 53

3.2 数据集应用案例 ······ 55
3.2.1 数据集条件过滤 ······ 55
3.2.2 CALL 子程序数据步应用 ······ 60
3.2.3 数据集输出应用 ······ 61
3.2.4 数据集加密码应用 ······ 64
3.3 外部数据处理案例 ······ 65
3.3.1 TXT 文件数据处理 ······ 65
3.3.2 Excel 数据处理 ······ 66
3.3.3 CSV 格式数据处理 ······ 68
3.3.4 DAT 格式数据处理 ······ 69
3.3.5 关系数据库数据处理 ······ 70
3.3.6 批量数据文件处理 ······ 72
3.3.7 宏过程数据处理 ······ 72
3.3.8 表格数据处理 ······ 73
3.3.9 二次数据处理 ······ 77
第 4 章 过程步基础与案例 ······ 79
4.1 过程步基础 ······ 79
4.1.1 过程步功能与定义 ······ 79
4.1.2 过程步应用 ······ 79
4.2 常见过程步应用 ······ 81
4.2.1 print 过程 ······ 81
4.2.2 means 过程 ······ 82
4.2.3 copy 过程 ······ 84
4.2.4 SQL 过程 ······ 85
4.2.5 report 过程 ······ 88
4.2.6 freq 过程 ······ 90
4.2.7 summary 过程 ······ 92
4.2.8 compare 过程 ······ 95
4.2.9 datasets 过程 ······ 96
4.2.10 surveyselect 抽样过程 ······ 98
4.2.11 format 过程 ······ 100
4.2.12 sort 过程 ······ 102
4.3 经济指数指标分析案例 ······ 104
第 5 章 函数基础与案例 ······ 106
5.1 函数基础 ······ 106
5.1.1 函数功能与常用函数 ······ 106
5.1.2 数据步引用函数 ······ 137
5.1.3 宏过程引用函数 ······ 137
5.1.4 函数综合应用 ······ 138

5.2 信用卡收入分析案例 ······ 139
第 6 章 宏基础与案例 ······ 142
6.1 宏基础 ······ 142
6.1.1 宏概述与定义 ······ 142
6.1.2 宏过程应用 ······ 147
6.2 文件夹判断案例 ······ 153
6.3 日期处理 ······ 154
6.4 批量读取同类文件 ······ 156
6.5 客户交易分析输出 ······ 157
6.6 批量文件压缩 ······ 158
第 7 章 统计分析基础与案例 ······ 161
7.1 统计分析基础 ······ 161
7.1.1 描述性统计过程概述 ······ 161
7.1.2 描述性统计过程应用 ······ 164
7.2 方差分析基础 ······ 174
7.2.1 方差分析概述 ······ 174
7.2.2 方差分析应用 ······ 177
7.3 相关分析与回归分析基础 ······ 179
7.3.1 相关分析与回归分析概述 ······ 179
7.3.2 相关分析与回归分析应用 ······ 185
7.4 因子分析基础 ······ 193
7.4.1 因子分析概述 ······ 193
7.4.2 因子分析应用 ······ 195
7.5 生存分析基础 ······ 197
7.5.1 生存分析概述 ······ 197
7.5.2 生存分析应用 ······ 200
7.6 聚类分析基础 ······ 204
7.6.1 聚类分析概述 ······ 204
7.6.2 聚类分析应用 ······ 207
7.7 判别分析基础 ······ 209
7.7.1 判别分析概述 ······ 209
7.7.2 判别分析应用 ······ 213
7.8 客户流失分析案例 ······ 216
第 8 章 SAS 与关系数据库 Oracle 交互应用 ······ 219
8.1 SAS 与 Oracle 交互基础 ······ 219
8.1.1 SAS 与 Oracle 数据库连接概述 ······ 219
8.1.2 SAS 获取 Oracle 数据 ······ 220
8.1.3 SAS 装载数据到 Oracle 数据库 ······ 223
8.1.4 Oracle 数据解数到外部数据文件 ······ 225

8.1.5 条件过滤取 Oracle 数据库中的数据 …… 228
8.2 信用卡交易流水数据提取案例 …… 230
第 9 章 ODS 基础与综合案例 …… 234
9.1 ODS 基础 …… 234
9.1.1 ODS 概述与功能 …… 234
9.1.2 ODS 定义与应用 …… 235
9.2 ODS 综合案例 …… 236
9.2.1 ODS 输出 PDF 文件 …… 236
9.2.2 ODS 输出 HTML 文件 …… 238
9.2.3 ODS 输出 CSV 格式文件 …… 240
9.2.4 ODS 与 Oracle 交互输出 PDF 文件 …… 242
9.2.5 ODS 输出 TXT 格式文件 …… 243
9.2.6 ODS 输出 DAT 格式文件 …… 244
9.2.7 ODS 输出 RTF 格式文件 …… 245
9.2.8 ODS 输出到打印机 …… 246

第 3 篇 综合实战篇

第 10 章 信用卡管理系统案例 …… 249
10.1 业务需求分析与架构设计流程 …… 249
10.2 ETL 层数据处理 …… 254
10.3 数据挖掘信贷风险案例 …… 281
10.4 SAS/EM 数据挖掘实现过程 …… 285
第 11 章 SAS 模型开发案例 …… 303
11.1 数据挖掘建模概述 …… 303
11.1.1 数据挖掘层级 …… 303
11.1.2 挖掘建模概念 …… 304
11.1.3 模型开发平台建设 …… 304
11.2 数据挖掘建模理论 …… 306
11.2.1 数据挖掘建模分类 …… 306
11.2.2 评分卡模型分类 …… 310
11.3 数据挖掘建模流程 …… 312
11.3.1 需求分析 …… 312
11.3.2 数据准备 …… 313
11.3.3 模型开发 …… 316
11.3.4 模型验证 …… 319
11.3.5 策略设计 …… 320
11.3.6 模型部署 …… 321
11.4 评分卡模型开发案例 …… 322
11.4.1 前段准备 …… 323

11.4.2 开发模型 …… 329
11.4.3 模型应用 …… 331
第 12 章 高级应用技巧 …… 332
12.1 自动变量与临时变量应用 …… 332
12.1.1 自动变量_N_与_ERROR_应用 …… 332
12.1.2 临时变量 FIRST.变量与 LAST.变量的应用 …… 335
12.2 SAS 索引应用 …… 337
12.2.1 索引简介 …… 337
12.2.2 索引的创建与删除 …… 340
12.2.3 索引的应用 …… 342
12.3 自定义 FORMAT 格式应用 …… 345
12.4 HASH 对象的应用 …… 348
参考文献 …… 355

第 1 篇　基　础　篇

第 1 章　SAS概述与整体架构

1.1　SAS概述

SAS（Statistical Analysis Software）曾是一个著名的统计软件，目前已经发展成为一个综合的组合应用软件系统。SAS 目前在金融、数据挖掘、医药、电信、统计学、商业智能（BI）、数学、教育机构、科研院所等领域得到了广泛的应用，其中 SAS 统计分析指标为国际标准。通过本章的学习，读者可以了解 SAS 语言和 SAS 工具组合软件的体系，把握 SAS 的整体架构。从而根据需要去学习 SAS 的特定模块。

1.1.1　SAS介绍

SAS 软件是在 20 世纪 70 年代由北卡罗来纳州立大学编写的，当时用来处理数据，做生物分析用。随着 SAS 的发展，SAS 研究院于 1975 年成立。SAS 公司逐渐发展壮大，现在已经成为世界最大的独立软件公司之一。目前最新版本是 9 系列。随着版本的发展，SAS 内部自带函数和过程一直在增加，功能逐渐强大和完善，图形化的操作也变得更灵活。

在最新版本 9 系列中，SAS 9.2 开始支持自定义函数，同时 SAS 也有了自己的数据库——SPDS 数据库，这是 SAS 软件的一大飞跃，也是 SAS 9 系列的一个亮点，显示了 SAS 的强大功能。

SAS 软件从 20 世纪 70 年代诞生到目前已经成为世界公认的国际标准软件。SAS 系统是把数学理论和实际应用结合起来的一个组合软件系统。SAS 的强大功能和应用领域的广泛性也促进了 SAS 的发展。SAS 语言和 SAS 工具将越来越强大，具有的模块和功能也越来越多，已经成为了一个综合的组合软件，应用领域从单一的统计分析，发展到目前的 OLAP 分析、数据挖掘、ETL 数据处理、SPDS 数据库、统计分析、数据分析等。

1.1.2　SAS行业应用现状

SAS 目前主要应用在金融、商业智能、数学、通信、经济、生物医药、教育机构和科研院所等领域。目前全世界很多国家都在使用 SAS。SAS 在金融领域可以用来做风险分析，信用评分、信用卡开卡的评分系统，底层数据抽取、转换与装载的工作（Extract，Transform and Load，ETL），处理大数据量的数据。在 BI（Business Intelligence，商业智能）方面 SAS 通过前端展现工具 SAS/SAS portal 用于报表的开发。SAS /EG 组件是 SAS 的一个客户端，可以连接 SAS 服务器，是开发 SAS 程序的客户端工具。

SAS 目前在统计分析领域是世界上公认的最具权威的统计分析工具，统计分析指标获得国际公认。

SAS 在金融领域可以对客户消费类型、信用评分、风险控制进行分析，挖掘潜在客户。

SAS 在数据挖掘领域已经开发出很多业界成熟的模型，如决策树、神经网络、逻辑回归等模型。

SAS 在商业智能方面，已经开发出 SAS portal 做报表开发。

SAS 目前在生物医药领域可以用来对生物医药研发的指标进行统计分析。

通过以上所列举的领域，可以看到 SAS 的应用领域非常广泛，发展前景有目共睹。SAS 工具集成了很多数学领域的公式，用户可以直接调用封装好的数学公式来进行指标的分析。只需要知道这些分析出来的指标代表什么意义，就可以根据这些指标写出分析报告，做出对未来的预测和决策支持。

1.2 SAS整体架构

SAS 已经发展成为一个综合的组合软件系统，理解并了解 SAS 系统的整体架构可以更清晰地认识 SAS 每一个模块具体实现的功能，同时更能深入了解各模块之间的联系和依赖关系。

1.2.1 SAS整体架构流程

SAS 整体架构流程如图 1-1 所示。SAS 平台架构主要由 4 层组成，分别是客户端层、中间层、服务层和数据层。中间的“元数据服务”是 SAS 系统对元数据的管理与维护，是其他 4 层的基础核心部分，与其他 4 层动态交互。

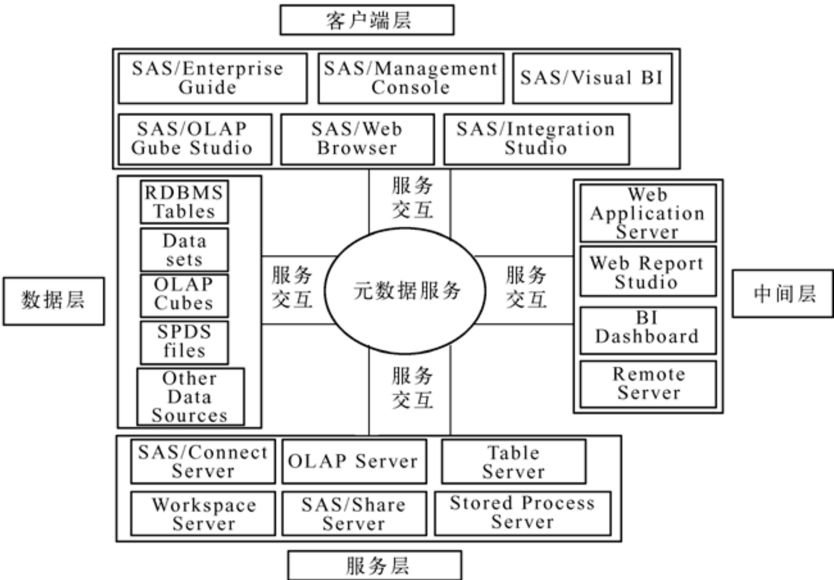


图 1-1　SAS 整体架构流程

1．客户端层

客户端层是用户实现与 SAS 服务器交互的前端工具，通过客户端工具操作 SAS 系统，从而利用 SAS 具有的服务。常用的 SAS 客户端工具如下。

1）SAS/EG（Enterprise Guide）：SAS 的前端工具，用户通过 SAS/EG 连接 SAS 服务器，如图 1-2 所示。

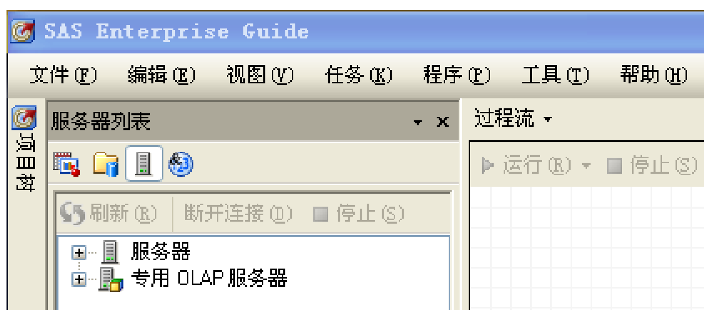


图 1-2　SAS Enterprise Guide

2）SAS/Management Console：SAS 对 SAS 服务器、SAS 逻辑库注册已经授权管理等可以通过此界面工具进行设置管理 SAS 用户和权限控制，如图 1-3 所示。

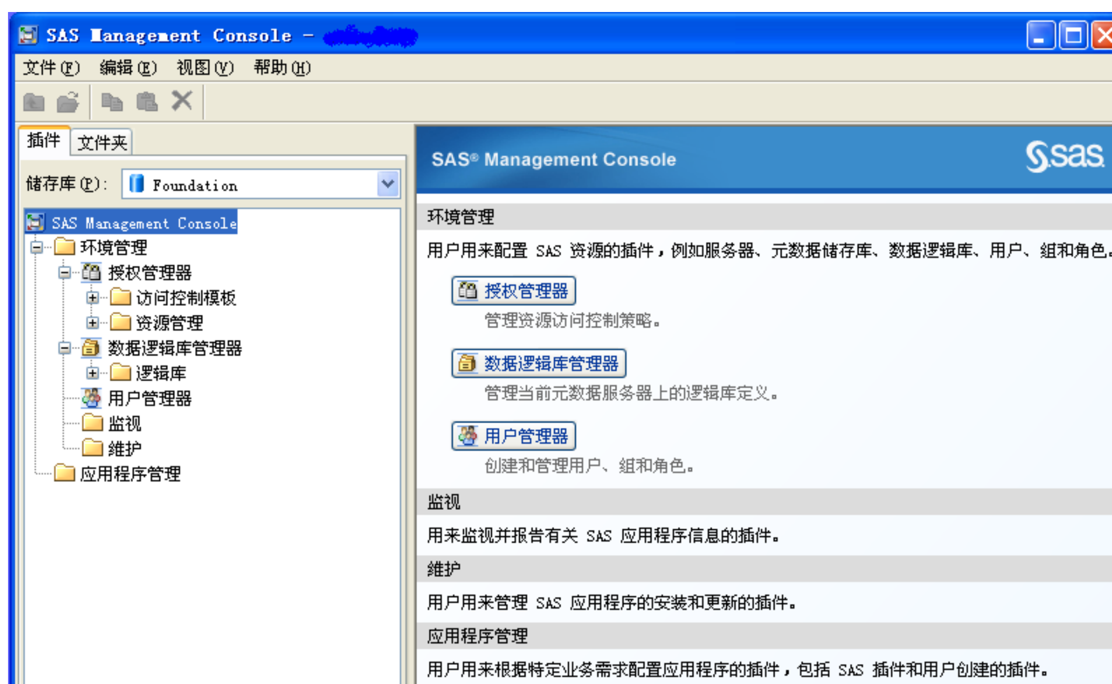


图 1-3　SAS Management Console

3）SAS/Visual BI：SAS 通过此类工具可以浏览商业智能所挖掘的信息，如 SAS/Data Mining（数据挖掘）。

4）SAS/OLAP Cube Studio：SAS 连接分析的客户端工具，如 SAS 报表开发、查询。

5）SAS/Web Browser：SAS 系统网页版的 Web 方式浏览结果展现的一种形式。

6）SAS/Integration Studio：SAS 系统对数据整合的工具。

2．数据层

数据层是 SAS 系统具有的数据格式，存储 SAS 能够处理的各种数据。常见的数据类型如下。

RDBMS Tables：SAS 系统对关系数据库数据的表信息存储，如 Oracle 数据库数据。

Data Sets：SAS 系统数据步生成的数据，称为 SAS 数据集。

OLAP Cubes：联机分析的多维立方体数据。

SPDS Files：SAS 具有的 SPDS 数据库生成的数据格式。

Other DATA Sources：SAS 系统具有的其他数据源，这里不再一一列举。

3．服务层

SAS 系统的服务层是对其他层提供服务支持的。常见的服务层如下。

SAS/Connect server：SAS 系统提供连接各服务器的服务，如连接外部数据文件或关系数据库的服务。

OLAP Server：SAS 系统提供联机分析的服务处理。

Table Server：SAS 系统对表信息提供的服务处理。

Workspace Server：SAS 系统对空间维护与分配提供的存储空间服务处理。

SAS/Share server：SAS 系统提供的共享服务处理。

Stored Process Server：SAS 系统提供的对各进程的服务。

4．中间层

中间层属于 SAS 系统与元数据层交互需要过渡的层，实现临时处理服务，使系统性能更高、更快。常见的中间层如下。

Web Application Server：SAS 系统为 Web 应用层服务提供的服务。

Web Report Studio：SAS 系统为报表展现提供的服务。

BI Dashboard：商业智能控制面板是向企业展示度量信息和关键业务指标（KPI）现状的数据虚拟化工具。控制面板在一个简单屏幕上整理数字、公制（metric）和绩效记分卡。通过调整适应特定角色并展示为单一视角或部门指定的度量指标信息。它具有动态交互功能，可以传输到 iPad 上展示。

Remote Servers：SAS 系统为远程其他的服务端提供服务。

1.2.2　SAS各模块实现功能

SAS 系统目前模块众多，已经发展成为一个功能强大的组合应用软性系统。每一个模块都是为满足用户不同的需求实现不同的功能而开发的，理解这些模块的功能可以帮助用户灵活选择模块，有针对性地应用和学习这些模块，而不是全部都去研究学习。SAS 系统常用模块及功能如表 1-1 所示。

表 1-1　SAS 常用模块及功能

SAS 主要模块	模 块 功 能
BASE SAS	SAS 系统的核心模块，是所有产品模块的基础，实现模块的调度、数据管理、数据访问，支持关系数据库 SQL 语言的处理，统计分析的基础和报表生成的基础模块
SAS/ACCESS	连接其他数据库的接口，如连接 Oracle 数据库、DB2、Sybase 和 Informix 等主流关系数据库，实现 SAS 与数据库的交互操作
SAS/ACCESS TO PC FILES	SAS 对外部文件的读取和生成各类型的文件需要调用此模块，如读取.dat 格式文件、.txt 格式文件、.csv 格式文件和 COBOL 语言生成的 IBM 主机文件等都需要调用此模块。通过此模块把文件转换成 SAS 能够识别的数据集，再运用其他模块进行分析、报表生成、作图等。该模块实现对文件的处理
SAS/EM	实现数据挖掘的功能，在这个模块里实现了 SAS 的 SEMMA 数据挖掘模式，同时还有可以直接应用的数据挖掘模型
SAS/STAT	统计分析应用模块，是统计学学习者应用的模块，主要运用在统计领域，如回归分析、相关分析、因子分析、聚类分析和判别分析等分析功能
SAS/EG	SAS 系统前端登录模块，可以编写 SAS 程序，实现与服务器的交互，属于一个图形化操作界面
SAS/Management Console	SAS 系统的 Management Console 模块属于一个图形化的模块，实现对 SAS 系统的管理、用户权限控制与分配、逻辑库管理等功能
SAS/Portal	SAS 系统的 Portal 模块是 SAS 系统实现报表开发的工具，通过此模块可以开发报表

第 2 章　SAS编程语法基础

2.1　逻辑库定义与应用案例

SAS 通过逻辑库建立与关系数据库或外部数据文件的连接。逻辑库是 SAS 与外部环境交互数据的通道。

2.1.1　逻辑库的作用与定义

1．逻辑库的作用

SAS 逻辑库是一个逻辑标识，真正对应的是指向物理文件路径，用来告诉 SAS 数据集或数据文件存储到什么位置的一个逻辑标识。通过逻辑库来告诉 SAS 系统数据集是存储在哪个文件夹下物理位置。引用对应逻辑库下的数据文件只需用逻辑库存储目录下的数据集名就可以直接引用。如果在生成一个数据集时没有指定逻辑库，SAS 默认存储在临时逻辑库 Work 下面，当 SAS 结束会话退出 SAS 系统时，Work 逻辑库的数据集会被自动删除，不保存数据集。人为建立的逻辑库是永久逻辑库，退出会话后永久逻辑库所对应目录下的数据集仍保留在对应物理位置。

SAS 逻辑库是 SAS 系统建立与外部关系数据库或外部数据文件交互的通道，通过逻辑库实现数据互通。逻辑库分为永久逻辑库和临时逻辑库。

永久逻辑库：此类逻辑库中对应的数据不会随着 SAS 的关闭与退出消失。

临时逻辑库：此类逻辑库中对应的数据会随着 SAS 的关闭或退出消失。

SAS 系统本身自带的逻辑库如下。

Sashelp 逻辑库：存储 SAS 帮助的数据集数据的永久逻辑库，只读逻辑库。

Sasuser 逻辑库：存储用户文件的逻辑库。

Work 临时逻辑库：存储临时数据集，退出会话后数据集不再存在。

为便于理解逻辑库的作用，首先在自己的计算机上运行如下 SAS 程序，带着对这个程序的疑问去理解和学习逻辑库。

【例 2.1】 创建客户信息数据集，存储到 d:\jx 文件夹目录下。

```
libname   jx   'd:\jx';   /*创建逻辑库 jx ，对应的物理位置为 d:\jx*/
data   jx.custer;   /*custer 数据集存储到 jx 逻辑库下  */
        input    id   name   $   sex   $   jf;
        CARDS;
        1000001  刘小华  M 100
        1000002  董大帅  F 3000
        1000003  郭美玉  M 6000
;
run;
```

```
proc   print   data=jx.custer   ; /*打印数据集输出到窗口*/
/*  引用逻辑库 jx 的 custer 数据集  */
run;/*结束标志*/
```

2．逻辑库的定义

SAS 逻辑库分为两类，一类是建立与外部数据文件连接的逻辑库，另一类是建立与关系数据库连接的逻辑库。

1）建立与外部数据文件或 SAS 数据集连接的逻辑库语法：

libname 逻辑库名‘物理路径’<选项>;

【语法解读】 libname 为建立逻辑库关键字，必选项。

逻辑库名为符合命名规范的名字。

‘物理路径’为数据文件存储路径。

<选项>为可选项，对逻辑库设置的选项，如 compress=yes、insertbuff=等。

【例 2.2】 通过 SAS 建立与 d:\jx 文件夹存储的数据逻辑库，逻辑库名为 jxsj。

```
libname   jxsj   'D:\jx'   compress=yes;   /*compress=yes 表示存储数据为压缩格式存储*/
```

2）建立与关系数据库连接的逻辑库语法：

libname 逻辑库名 库引擎 user=数据库用户名 password=数据库密码 path=数据库实例<选项>;

【语法解读】 库引擎为关系数据库引擎，如 Oracle、Teradata、DB2、MySQL、SASSPDS 等关系数据库引擎。

user=数据库用户名：登录关系数据库的用户名。

password=数据库密码：登录关系数据库的密码。

path=数据库实例：登录关系数据库的实例。

【例 2.3】 建立 SAS 与关系数据库 Oracle 的库连接，逻辑库名为 jxlk，登录个人 Oracle 数据库。本机登录 Oracle 数据库用户名为 chiran，密码为 chiran，数据库实例为 orcl，具体实验时要修改自己本机的库连接信息。

```
libname   jxlk   oracle   user=chiran   password=chiran   path=orcl;
```

【语法解读】jxlk 逻辑库作为一个桥梁，建立 SAS 与 Oracle 数据库连接的通道，使 SAS 数据与 Oracle 数据库中的数据可以互相交换。数据库中的表直接通过逻辑库就可以引用到 SAS 环境中来。

【例 2.4】 SAS 连接自己的 spds 数据库，建立逻辑库名为 cr。

```
libname   cr   sasspds   'ycr'   host='主机名'   service='服务器端口号'   user='用户名'   password='密码';
```

【说明】 SASSPDS 库引擎是 SAS 自己的数据库。

3．引用逻辑库

引用逻辑库是对创建好的逻辑库所对应的数据文件或数据集，或外部数据库数据的数据应用。

建立逻辑库的目的就是引用逻辑库所对应的数据。在数据步或过程步中通过逻辑库来调用所需要的数据集。

引用逻辑库语法：逻辑库名.数据集名。

【注意】 中间用英文状态下的点分割。引用非临时逻辑库的数据集使用两级命名方式，引用 Work（临时逻辑）的数据集时，可以直接使用数据集名。

【例 2.5】 打印输出 jx 逻辑库中的数据集 custer。

```
Proc  print  data=jx.custer  ; /*jx 逻辑库名，custer 是 jx 逻辑库下对应的数据集名*/
/* 引用逻辑库 jx 的 custer 数据集 */
run;/*结束标志*/
```

2.1.2 Windows环境与UNIX环境创建逻辑库

SAS 服务器安装在 Windows 服务器上时，其创建逻辑库时需要注意书写物理路径，与 UNIX 环境书写物理路径不同，对比学习，可以看到它们之间的差异。对于与外部数据文件存储目录建立逻辑库而言，其书写物理路径方式不同。

1．Windows环境

服务器物理路径写法如下:

盘符:\文件夹路径

【例 2.6】 建立 SAS 与 d:\jx 路径下的文件夹逻辑库。

```
libname  jx  'd:\jx';
```

2．UNIX 环境

服务器物理路径写法如下：

/文件夹路径

【例 2.7】 建立 SAS 与/home/db/test 路径下的文件夹逻辑库。

```
libname  ujx  '/home/db/test';
```

通过上面的案例可以看出，Windows 服务器环境与 UNIX 服务器建立逻辑库的差异主要是书写数据存储物理路径的方式不同。

2.1.3 逻辑库与关系数据库的连接方式

逻辑库与关系数据的连接方式主要有以下两种。

1．宏变量书写方式连接关系数据库。

【例 2.8】 Oracle 数据库，作者个人本机登录 Oracle 数据库的用户名为 chiran，密码为 chiran，数据库实例为 orcl，通过 SAS 逻辑库建立 SAS 与 Oracle 数据库的连接通道。

```
libname  jx  oracle  user=chiran  password=chiran  path=orcl;
```

2．SQL过程连接关系数据库。

【说明】 SQL 过程是 SAS 系统自带的过程，用户可以直接调用。

【例 2.9】 SQL 过程与宏变量结合应用连接关系数据库，对例 2.8 进行改造。

```
%let  jx_connect=user=chiran  password=chiran  path=orcl;  /*定义连接数据库的宏变量*/
proc  sql  noprint;  /*调用 SAS 内部 SQL 过程*/
```

```
connect to oracle (&jx_connect);     /*取连接数据库的宏变量 jx_connect*/
        select * into :v_sj from connection to oracle
                        ( select to_char(bksj,'yyyymmdd') from sj);
        Execute (create table jxc as select * from sj) by oracle;
            disconnect from oracle;
quit;
```

【程序解读】

1）%let jx_connect=user=chiran password=chiran path=orcl：定义连接数据库宏变量语句，通过此语句把数据库的连接信息赋值给宏变量 jx_connect。

2）connect to oracle (&jx_connect)：通过 connect to oracle 固定语句建立连接 Oracle 关系数据库的连接方式，&jx_connect 取连接 Oracle 数据库的连接信息。

3）select * into：v_sj from connection to oracle

(select to_char(bksj,'yyyymmdd') from sj);

execute (create table jxc as select * from sj) by oracle;

上面的语句为 SQL 过程中可执行的 SQL 语句。

4）disconnect from oracle：固定语句，断开与 Oracle 数据库的连接。

5）quit：SQL 过程的结束标志。

【例 2.10】 对例 2.8 进行改造，通过 SQL 过程连接数据库，查询数据库表 sj ，并把表字段 bksj 数据赋值给 v_sj，通过此过程创建一个新表 jccre。

```
proc sql noprint;
     connect to oracle (user=chiran password=chiran path=orcl);
      select * into :v_sj from connection to oracle
          ( select to_char(bksj,'yyyymmdd') from sj);
      execute (create table jxcre as select * from sj) by oracle;
       disconnect from oracle;
quit;
```

【程序解读】 connect to oracle (user=chiran password=chiran path=orcl)：此语句括号内直接写连接数据库的登录用户信息，对比例 2.8 宏变量的方式，可以看出这是两种不同的书写方式，具体选择哪种方式需要根据实际开发确定。两种方式都能实现关系数据库的连接，但宏变量的方式更方便管理和修改，尤其是需要经常更改用户登录关系数据库信息的地方。

2.2 SAS编程语法

学习 SAS，首先需要掌握这种编程语言的编程语法。SAS 编程语言是 SAS 编程的基础，是其他综合开发应用的基础。把 SAS 编程语句融合在应用案例中详细讲解，更能把理论和实践结合起来。在实践中去学习 SAS 编程语言也是最快最有效地掌握 SAS 的一种学习方式，同时能体验到业界真正应用的知识点。学好本章也是学习后面经典案例、统计分析、数据挖掘和模型开发的基础。

2.2.1 变量与常量

1. 变量

变量是指没有固定的值，可以改变的数。变量用于描述某字段的属性。例如，一个人具有年龄，年龄就属于一个变量。对于一个 SAS 数据集或关系数据库中的表而言，表中定义的列中的变量是有属性的，如字符、数值、日期等。SAS 中的变量有两种类型：数值型和字符型。SAS 中的变量在使用时不需要定义，它属于弱类型的，具体类型依赖赋值类型。

（1）变量命名规范

SAS 中变量的命名规范是以字母（a，b，c，…，z）或下画线“_”开始，字母不区分大小写，后面的字符可以是数字、字母或下画线。不能在定义变量中使用系统保留的名称，（如“_all_”、“_N_”、“_ERROR_”、“_INFILE_”和“_CHARACTER_”等），不能使用特殊字符（如%、&、#）和空格，最长 32 个字符。

下面举几个例子帮助读者理解命名规范。

【例 2.11】 变量命名举例。

正确的命名：z、_y、k_。

错误的命名：%x、&。

（2）变量赋值

SAS 编程中给变量赋值的一种方式是在数据步中通过 INPUT 语句将外部文件中的数据、CARDS 或 DATALINES 后面的输入数据赋值给变量。也可以在数据步中直接给变量赋值。如果是外部存在的数据文件，读入时用 INFILE 语句获取外部数据文件，但字段变量要在 INPUT 语句中定义。另一种方式是通过宏变量直接给变量赋值。

【注意】 INPUT 语句读入的字符变量默认长度为 8 个字节，超过 8 个字节就要使用 LENGTH 语句先定义变量并指明其长度。

【提示】 INPUT 语句和 LENGTH 语句中定义的变量如果是字符型的要加$。

【例 2.12】 数据步中定义变量 x 和 y，并给其赋值。

```
DATA   bl;
x=2;        /*数据集中定义变量 x，是数值型*/
y='abcd';   /*数据集中定义变量 y，是字符型*/
RUN;
```

【例 2.13】 读取客户身份证号信息和性别的数据，建立数据集 custer。

```
DATA   custer;
length   sf   $18.;   /*数据集中定义身份证号变量 sf，是字符型的加$符号*/
input    sf   sex   $;   /*sex 变量类型为字符类型，加$符号，取默认长度 8 字节存储*/
CARDS;
170826186532435435   M
210356789235875763   F
334567889123445566   M
434567789009898753   F
;
RUN;
```

【程序解读】

1）由于身份证号变量 sf 的长度超过 SAS 默认的长度 8B，需要首先通过 length 语句定义变量 sf。

2）INPUT 语句中直接引用定义的变量 sf，并可以直接定义变量 sex，然后用 PDV 指针控制读取 CARDS 语句中的数据。

【例 2.14】 直接给宏变量赋值。

```
%let  wj='d:\jx\stu.txt';
/*宏变量 wj 直接赋值，把路径文件 d:\jx\stu.txt 赋值给宏变量 wj*/
```

【程序解读】

%let 语句定义宏变量 wj，这个变量存储的值为外部文件存储位置的物理路径。

（3）变量类型转换规则

SAS 对于字符型与数值型之间自动转换的规则如下：

1）字符型转换为数值型。

- 字符型变量和数值型变量做运算或比较运算时，字符型变量自动转换为数值型变量。
- 字符型变量赋值给数值型变量时，字符型变量自动转换为数值型变量。

【例 2.15】 字符 y 变量自动转换为数值型，与 x 做加法运算。

```
data  qsum;
x=1;          /*定义为数值型变量 x*/
y='2';        /*定义为字符型变量 y*/
sum=x+y;      /*y 字符型变量自动转换为数值型做运算*/
run;
```

【程序解读】 由于 SAS 属于弱类型语言，y 变量为字符类型，引用 y 与 x 变量一起做加法运算时自动转化为数值类型，并把求得的和赋值给 sum。

2）数值型转换为字符型。

- 数值型变量赋值给字符型变量时，数值型变量自动转换为字符型变量。
- 数值型变量与字符型变量做字符连接运算时，数值型变量自动转换为字符型变量。
- 数值型变量用在字符处理函数中时，数值型变量自动转换为字符型变量。

【例 2.16】 数值型变量 y 与字符型变量 ch 做连接运算。

```
 data  substr;
y=10;           /*定义为数值型变量 y*/
ch='hellow';    /*定义为字符型变量 ch*/
v_substr=ch||y; /*v_substr 数值型变量自动转换为字符型变量做连接运算*/
run;
```

【程序解读】 数据步中变量 y 为数值类型，变量 v_substr 的值为 ch 变量与 y 变量的连接串的值，此处 y 变量自动转换为字符类型。

【例 2.17】 用字符型函数把数值型变量自动转换为数值型。

```
data   ytochar;
y=62345679;        /*定义为数值型变量 y*/
c=substr(y,5,6);   /*字符型函数截取数值型变量自动将 y 转换为字符型变量*/
```

```
run;
```

【程序解读】 数据步中变量 y 为数值，变量 c 中的值为通过字符处理函数 substr 截取的 y 变量的值，此处 y 自动转化为字符类型引用。

【提示】 此处 y 在字符函数中被转换为 BEST12.的输出格式。字符值右对齐，相当于左边补空格，补到 12 位。

2．常量

常量是其值不可变化的量。常量是相对变量而言的，变量可以重新赋值，而常量是固定不变的值。SAS 中定义的常量与其他语言中定义的常量基本相同。

SAS 中经常用的常量类型有 3 种：字符型常量、数值型常量和日期型常量，十六进制常量不经常用。

（1）字符型常量

1）一般字符常量用单引号或双引号括起来都可以，效果一样。

【例 2.18】 常量赋值给变量。

```
data   cl;
clz='跟案例学 SAS';   /*常量用单引号括起来*/
cl="what do you say?";     /*常量用双引号括起来*/
run;
```

【程序解读】 数据步中的 clz 变量用单引号括起字符串，cl 变量用双引号括起字符串，对比输出信息可以看到两种方式都可以处理字符串，效果一样。

2）对于字符常量中包含特殊字符的常量对单引号和双引号的使用有要求。如果字符常量中包含单引号外面就用双引号括起来，如果字符常量中包含双引号外面就用单引号括起来。

【例 2.19】 包含单引号和双引号的两个常量赋值给变量。

```
data   yk;
sstr="Tom's";     /*常量中包含单引号，外面用双引号括起来*/
dstr='学习者"灵活"体会应用案例学习方法';   /*常量中包含双引号，外面用单引号括起来*/
run;
```

【程序解读】 数据步中的 sstr 变量中的赋值字符串中包含单引号，要使单引号包含在字符串中，外面用双引号括起来；dstr 变量字符串赋值中包含双引号，要使双引号包含在字符串中，外面用单引号括起来。

（2）数值型常量

数值型常量就是类型为数值的常量。如果数值型常量大于 10E32−1，必须用科学计数法表示。

【例 2.20】 数值型常量赋值给变量。

```
data   cl;
x=78;       /*数值型常量赋值给变量 x*/
y=−121;     /*负数值型常量赋值给变量 y*/
z=+39;     /*正数值型常量赋值给变量 z*/
k=2E−3;     /*科学计数法数值型常量赋值给变量 k*/
run;
```

【程序解读】 数值常量直接赋值给变量，变量不需要定义。

（3）日期型常量

日期型常量包括日期（date）、时间（time）和日期时间（datetime）3 种类型。这 3 种类型的常量都要用单引号或双引号括起来。如果是日期型的常量，后面加字母 d；如果是时间型的常量，后面加字母 t；如果是日期时间型的常量，后面加字母 dt。

【例 2.21】 日期、时间和日期时间类型的 3 个常量分别赋值给变量。

```
 data    dttime;
cdate="3mar2012"d;      /*日期型常量赋值给变量 cdate*/
ctime='8:30't;       /*时间型常量赋值给变量 ctime*/
cdatetime='6apr2012:8:18:30pm'dt;;     /*日期时间型常量赋值给变量 cdatetime*/
run;
proc    print;/*调用打印过程打印输出显示信息到输出窗口*/
format    cdate    yymmdd10.    ctime    time10.    cdatetime    datetime22. ;
/*定义时间的输出格式  */
run;
```

【程序解读】 对于日期、时间和日期时间类型的值，赋值数据带标志符号。本实例中的 cdate 变量为日期型，后面加 d（date）表示为日期类型；ctime 变量为时间型，后面赋值数据加 t（time）表示为时间类型数据；cdatetime 变量为日期时间型，赋值数据后面加 dt（datetime）表示日期时间类型。

输出窗口显示结果，如图 2-1 所示。

```
输出 － （无标题）
                         SAS 系统          2012年04月06日 星其
Obs        cdate        ctime            cdatetime
  1    2012-03-03      8:30:00      06APR2012:20:18:30
```

图 2-1 日期时间常量赋值给变量的输出显示窗口

（4）十六进制常量

十六进制常量是计算机计数法中以十六进制计数的一种方式，由偶数个数组成。它是数值型常量的另一种进制的特殊计数方式。十六进制数用单引号括起来，外面加一个 X 字母，这是十六进制的标志，如'1F'X。

2.2.2 条件选择语句与循环语句

对于编程语言而言每一种语言都有条件选择语句和循环语句，SAS 编程语言也不例外。SAS 通过 SAS 语言对数据进行处理，生成有效合理的数据集，便于其他模块用来进行分析、数据挖掘和模型开发，而生成数据集质量的差异关键在于处理数据。数据步中通过选择语句、循环语句、内部函数以及其他控制语句来处理数据集。其他语言的选择语句和循环语句与 SAS 语言的选择语句和循环语句差不多。

1．选择控制语句

（1）IF 语句

语法格式：IF 条件表达式 THEN 执行语句;

<ELSE 执行语句;>

【语法解读】

IF：选择语句关键字。

条件表达式：可以取比较运算符号组成的语句或逻辑运算符号组成的语句。

THEN：选择语句关键字，若条件表达式的条件成立则执行 THEN 语句后面的语句。

<ELSE 执行语句;>：可选项，如果 IF 语句条件不成立，有 ELSE 语句就执行 ELSE 语句后面的语句。

【例 2.22】 IF 语句案例应用：根据信用卡数据查找出各类卡的信息，并打印输出 C 类信用卡的信息。

```
 data    custer;
      input    id   $   level   $   amount   5.   @;
      label   id='卡编号'   level='级别'   amount='信用额度';
cards;
1001 A 20000
1002 B 3000
103 C 5000
105 A 90000
106 A 70000
107 A 50000
108 C 2000
;
data    atype    btype    ctype;
set    custer;
if   level='A'   then   output   atype ;    /*A 类卡输出到数据集 atype*/
    else   if   level='B'   then   output   btype;   /*B 类卡输出到数据集 btype*/
else   output   ctype;   /*C 类卡输出到数据集 ctype*/
run;
proc   print   data=ctype   label;   /*打印输出 C 类卡,这里要显示标签内容要加 label 参数*/
title "c 类信用卡额度";
run;
```

【程序解读】

1）IF 语句根据比较运算符号组成的语句判断，若条件 level='A'成立，则执行 THEN 语句，把符合条件的数据通过 OUTPUT 语句输出到数据集 atype 保存。

2）ELSE IF 语句对比较运算符号组成的语句进行判断，若条件 level='B'成立，则执行 THEN 语句，把符合条件的数据通过 OUTPUT 语句输出到数据集 btype。

3）对于 IF 语句与 ELSE IF 语句都不成立的，执行 ELSE 语句，通过 OUTPUT 语句把数据输出到 ctype 数据集中。

输出窗口显示输出结果，如图 2−2 所示。

输出 – （无标题）

c类信用卡额度　　2012年

Obs	卡编号	级别	信用额度
1	103	C	5000
2	108	C	2000

图 2-2　C 类信用卡输出显示窗口

（2）IF 条件表达式语句

语法格式：IF 比较运算符号组成的语句。

【语法解读】 比较运算符号组成的语句：比较条件判断语句。

功能：对数据集进行过滤。

【例 2.23】 IF 条件表件表达式语句应用，从数据集取 3 条记录。

程序如下：

```
libname  jx   'd:\jx';  /*定义逻辑库，指向物理路径为 d:\jx*/
 data   jx.credcard;
   input   id  $  level  $  amount  5.  @;
 cards;
1001 A 30000
1002 B 4000
1003 C 2000
1005 A 80000
1006 A 90000
1007 A 60000
1008 C 7000
;
data   filter3;
  set    jx.credcard;
  if   _N_<4;  /*从数据集取 3 条记录*/
run;
```

【程序解读】 对 SET 语句中的数据集 credcard 进行处理，通过 IF 语句引入 SAS 系统内部变量_N_。该变量值为数据步中读取记录条数的值，通过小于比较运算符号与 4 比较，如果_N_<4 成立就从数据集 creditcard 取出符合条件的数据，存储到数据集 filter3 中。

（3）SELECT 语句

语法格式：SELECT <查询表达式>;

WHEN (条件表达式) 执行语句;

<…WHEN (条件表达式) 执行语句;

<OTHERWISE 执行语句>;

【语法解读】

SELECT：语句关键字。

查询表达式：可选语句，一般为数据集中的一个变量。省略此项时，查询条件在 WHEN

语句中写明。

WHEN：语句关键字。

OTHERWISE：可选语句，在上面条件都不成立时执行 OTHERWISE 语句中的语句。

【例 2.24】 例 2.22 用 IF 语句实现的程序可以改造成用 SELECT 语句实现，两个程序实现的功能相同。

程序如下：

```
libname   jx    'd:\jx';/*定义逻辑库，指向物理路径为 d:\jx*/
   data     jx.credcard;
         input    id   $   level   $   amount   5.   @;
      cards;
    1001 A 30000
    1002 B 4000
    1003 C 2000
    1005 A 80000
    1006 A 90000
    1007 A 60000
    1008 C 7000
   ;
  data   atype   btype   ctype;
      set   jx.credcard;
      select   (level);     /*对 level 字段选择查询*/
when   ('A') output   atype;     /*条件若成立则输出到 atype 数据*/
when   ('B')   output   btype;     /*条件若成立则输出到 btype 数据*/
otherwise   output   ctype;      /*上面条件都不成立则输出到 ctype 数据'*/
end;
  run;
```

【程序解读】 SELECT 语句中查询 level 变量，与 WHEN 语句中的值比较判断，如果符合条件，就执行 WHEN 语句中的语句；否则执行 OTHERWISE 语句中的语句。

【例 2.25】 SELECT 语句对数据步中的变量处理应用，根据条件把变量 zone 的内容修改成所对应的城市。通过 SELECT 语句，用以下两种写法实现。

程序 1：

```
data   zonetocity;
    input   zone   $   address   $   @@;
  cards;
100070  丰台区  272195  微山县  200000  北京
;
data   changezone ;
     set    zonetocity;
     select;    /*无查询条件*/
       when   (zone='100070')   zone='北京'; /*条件语句写在 when 语句中，若成立执行 zone='北京'*/
       when   (zone='020')   zone='山东';    /*条件若成立执行 zone='山东'*/
       otherwise   zone='其他省份';          /*上面条件都不成立执行 zone='其他省份'*/
    end;
```

```
run;
```

程序 2：

```
data   changezone2 ;
  set     zonetocity;
    select   (zone);     /*无查询条件*/
      when    ('100070')      zone='北京';     /*条件语句写在 when 语句中，若成立执行 zone='北京'*/
      when    ('020')      zone='山东';          /*条件若成立执行 zone='山东'*/
        otherwise    zone='其他省份';       /*上面条件都不成立执行 zone='其他省份'*/
    end;
run;
```

【提示】 这两种 SELECT 语句虽然写法不同，但可以实现同样的功能。

2．循环控制语句

（1）DO WHILE 循环语句

语法格式：DO WHILE (条件表达式);

执行的 SAS 语句;

END;

【例 2.26】 求 x 自动加 2 的和。

```
  data    autoplus;
        x=0;
       do    while (x<11);      /*条件成立执行下面的语句*/
        x=x+2;
        end;
    ;
put    x=;    /*输出到日志窗口*/
run;
```

【程序解读】 循环条件是 x<11，当 x=10 时，x=x+2=12，此时返回到循环条件判断，已经不符合条件，循环结束。

日志中显示结果：x=12，如图 2-3 所示。

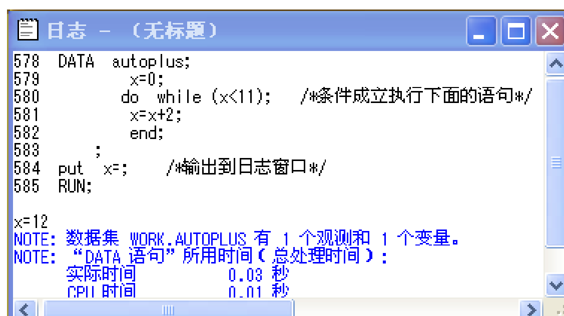


图 2-3　DO WHILE 循环日志输出显示窗口

（2）DO　UNTIL 循环语句

语法格式：DO UNTIL (条件语句)

执行的 SAS 语句;
END;

【注意】 DO UNTIL 循环语句，先执行后判断条件，条件成立输出结果。

【例 2.27】 DO UNTIL 循环语句应用。

程序如下：

```
data  un;
  x=8;
  do  until  (x=10);  /*先执行后判断，条件成立输出结果*/
    x=x+2;
  end;
put  x=;   /*输出结果到日志窗口*/
run;
```

日志结果显示 x=10。

（3）DO TO 循环语句

【语法格式】 DO 开始变量值 TO 终止值 <BY 每次增加量> ;

【语法解读】 BY 每次增加量：BY 语句中指定循环增量，默认为 1，省略此项时取默认 1 递增。

【例 2.28】 DO TO 循环语句应用，每次 x 值加 3。

```
data  increase;
  do  i=1  to  9  by  3  ;  /*每次 i 递增 3 个值*/
     x=i;
  end;
  put  x=; /*输出结果到日志窗口*/
run;
```

日志结果显示 y=7。

2.2.3 操作符与宏变量

1. 操作符

操作符运算是每种编程语言基本都具有的。SAS 语言的操作符包括算术运算符、比较运算符、逻辑运算符和字符串连接符等四大类。

（1）算术运算符

算术运算符号实现对数值类型的数据进行算术计算。常用算术运算符号如表 2−1 所示。

表 2−1 常用算术运算符号

算术运算符号	功 能
+	加法运算
—	减法运算
*	乘法运算
* *	乘方运算
/	除法运算

【例 2.29】 求 2 的三次幂，并求出 4 除以 2 的值。

```
 data   czf;
y=2**3; /*通过乘方运算符实现了 2 的三次幂*/
k=4/2;/*通过除法运算符实现了 4 除以 2 的运算*/
run;
```

（2）比较运算符

比较运算符号实现比较运算。常用比较运算符号如表 2-2 所示。

表 2-2 常用比较运算符号

比较运算符号	功 能
=或 EQ	相等条件判断
^=或 NE	不相等条件判断
>或 GT	大于条件判断
<或 LT	小于条件判断
>=或 GE	大于等于条件判断
<=或 LE	小于等于条件判断

【例 2.30】 比较运算符号“>=”应用，根据信用卡信用积分查找出信用积分小于 300 的客户信息。

```
data   cred_score;
          input   id   $   name   $   sex   $   score   @@;
cards;
   1001  张小红  F 300 1002  高峰名  M 700 1003  刘心洪  F 900 1005  赵用武  M 200
;
data   great300   littel300;
set   cred_score;
if   score>=300   then   output   great300;   /*信用积分大于或等于 300 的输出到数据集 great300*/
     else   output   littel300;        /*信用积分小于 300 的输出到数据集 littel300*/
run;
proc   print   data=littel300;          /*打印信用积分小于 300 分的客户信息*/
           title     "信用积分小于 300 分的客户";   /*打印输出标题*/
run;
```

【程序解读】 IF 语句中对成绩变量 score 通过比较运算符号>=把成绩大于或等于 300 分的数据信息输出到数据集 great300，不及格的执行 ELSE OUTPUT 语句输出到数据集 littel300。

输出窗口显示信用积分 littel300 的数据集信息，输出结果如图 2-4 所示。

图 2-4 信息积分小于 300 分的客户

（3）逻辑运算符

逻辑运算符号实现逻辑运算，计算结果只有两种值：真或假。如果逻辑值为真，返回计算结果为 1；否则为假返回计算结果为 0。常用逻辑运算符号如表 2-3 所示。

表 2-3　常用逻辑运算符号

逻辑运算符号	功　能
AND 或&	且，若判断条件中两个条件都为真，则值为 1，否则为 0
OR 或 |	或，若判断条件中一个为真或都为真，则值为 1，否则为 0

1）AND 或&。

【例 2.31】 x>y and y>0，如果 x>y 为真，y>0 为真，则结果为 1，否则有一个为假的结果就为 0。

2）|或 OR

【例 2.32】 x>y|y>0，如果两个关系中一个或两个都为真，则结果为 1；全为假则结果为 0。

（4）字符串连接符

字符串连接符号实现字符串的拼接或变量与变量的拼接，组合成新变量对应的内容。

功能：连接两个或多个字符串。

常用字符串连接符：||或!!

【例 2.33】 字符连接运用。

```
    data   cstr;
           str1='what';
           str2=' do you say!';
           cstr1=str1||str2;       /*用||实现字符串拼接连接组合成新串 cstr1*/
           cstr2=str1!!str2;       /*用!!实现字符串连接组合成新串 cstr2*/
run;
proc   print   data=cstr;
run;
```

【程序解读】 数据步中 cstr1 变量为字符串变量 str1 与 str2 通过字符串连接符号“||”实现字符变量的拼接后赋值的字符串；cstr2 变量为字符串变量 str1 与 str2 通过字符串连接符号“!!”实现字符变量的拼接后赋值的字符串。对比这两个字符串连接符号，可以看出功能一样，只是写法不同。

输出窗口显示输出结果，如图 2-5 所示。

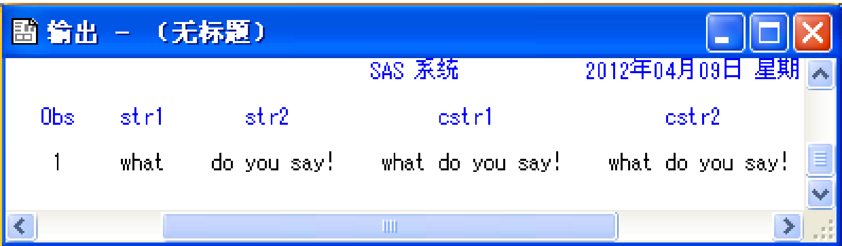


图 2-5　字符串连接信息输出显示窗口

2．宏变量

宏变量与其他语言中定义一个变量的意义一样，在程序的其他地方可以引用它。

定义宏变量的语法格式：%let 宏变量名<=赋值>;

【语法解读】

%let ：定义宏变量关键字。

宏变量名：定义的变量名，必须符合变量名定义规则。

=赋值：可选项，可以赋值，也可以通过其他语句赋值。

【注意】 一个宏变量语句中如果定义多个宏变量，宏变量之间用空格分隔。

【例 2.34】 定义一个不赋值宏变量 v_defin。

```
%let  v_defin;
```

【例 2.35】 定义一个赋值宏变量 v_dir，赋值为指向的路径值“d:jx”。

```
%let  v_dir='d:\jx';
```

2.2.4 格式修饰符与指针控制

1．格式修饰符

在进行实际 SAS 应用开发时，在数据步中对数据的处理中会遇到一些数据文件或数据块中数据包含特殊字符，对这样的数据在建立数据集读取数据时要在 INPUT 语句中定义变量的类型前加格式修饰符，这样才能读取这类数据文件或数据块数据。

常见的格式修饰符号有以下几种。

1）:（冒号）格式修饰符：从非空格开始读取变量所对应的数据，直到满足以下任何一种情况。

- 遇到下一个空格列。
- 对应变量所定义的长度已经读满。
- 数据行结束。

【提示】 对于上面的 3 种情况只要满足其中的一种情况该字段列读取的数据就结束。对于字段列对应的数据长度大小不定的加冒号修饰符号，可以正确读取数据，防止错读列。

【例 2.36】 冒号格式修饰符应用：通信地址最长 20 个字符，通过格式修饰符正确读取下面的数据。

```
data  address;
     input   name  $  tx_address  :$20.  zone  $  @;
     /*tx_address 列加了冒号格式修饰符，防止长度不够从下一列读取*/
   cards;
 张华  中国北京市丰台区  100070
 刘小峰  北京西城区  100000
 高兴于  中国山东  272195
     ;
run;
proc  print;
```

```
run;
```

【程序解读】 INPUT 语句中定义的变量 tx_address 所对应的变量类型前加冒号，告诉此变量如果长度不够遇到空格就结束读取；如果超过此长度也结束，数据行结束也结束读取。对于上面的程序，如果变量 tx_address 去掉冒号格式修饰符，数据读取就会错位。

输出窗口显示输出结果，如图 2-6 所示。

```
输出 - （无标题）
1    张华      中国北京市丰台区      100070
2    刘小峰    北京西城区            100000
3    高兴于    中国山东              272195
```

图 2-6　冒号修饰 tx_address 读取数据输出显示窗口

2）&格式修饰符：修饰所读取为字符型的列数据中含有一个或一个以上空格的字符数据。SAS 语言中默认空格为字段分隔符，如果想保留空格，必须在定义的字符列加格式修饰符，而它后面列的数据必须以两个或两个以上的空格分割。

【例 2.37】 &符号应用：读取通信地址含空格的数据。

```
data  address_blank;
   input   name  $  tx_address  &:$20.  zone  $6.  @;
     /*tx_address 列加了冒号格式修饰符，防止长度不够从下一列读取*/
   cards;
 张华  中国  北京市  丰台区   100070
 刘小峰  北京市  西城区   100000
 高兴于  中国  山东   272195
       ;
run;
proc  print;
run;
```

输出窗口显示输出结果，如图 2-7 所示。

```
输出 - （无标题）
                    SAS 系统          2012年04月09E

Obs    name      tx_address              zone

 1     张华      中国 北京市 丰台区       100070
 2     刘小峰    北京市 西城区            100000
 3     高兴于    中国 山东                272195
```

图 2-7　取符号修饰 address 读取字符列数据输出显示窗口

3）~ 格式修饰符：修饰所读取对应列包含单引号、双引号或分隔符的字符列。

【例 2.38】 ~符号应用：通信地址字段包含单引号或双引号，或分隔符号的字符数据。

```
data  ts;
```

```
            input   name   $   tx_address   ~&:$26.   zone   $6.   @;
        /*address 列加了~读取包含单引号或双引号，或分隔符号的字符数据*/
       cards;
      张华  "中国  ,北京市, 丰台区"   100070
      董小青  "北京  东城区"   100000
      刘小峰  "山东省  济南市"   270000
      高明明   '北京市, 西城区'   100000
      张东杨   '四川  成都'   345678
      王霞会   江苏, 南京   123456
        ;
run;
proc   print;
run;
```

【程序解读】 INPUT 语句中定义的变量 tx_address 中加~格式修饰符号，因为此列对应数据有单引号和双引号。

输出窗口显示输出结果，如图 2-8 所示。

输出 － （无标题）

Obs	name	tx_address	zone
1	张华	"中国 ,北京市, 丰台区"	100070
2	董小青	"北京 东城区"	100000
3	刘小峰	"山东省 济南市"	270000
4	高明明	'北京市, 西城区'	100000
5	张东杨	'四川 成都'	345678
6	王霞会	江苏, 南京	123456

图 2-8 ~ 修饰 tx_address 读取字符列数据输出显示窗口

2. 指针控制

SAS 语言有自己的指针控制，SAS 在读取数据时是通过 SAS 的指针控制符来控制读取数据的。指针控制符分为行指针和列指针两种，下面分别对这两种指针控制符在数据处理中的应用给予详细讲解。

（1）列指针控制符模式

语法格式：@n。

【语法解读】

@：列指针标志符号。

n：整数类型，指明列的开始位置，是对应变量的数据开始列位置。

【例 2.39】 列指针控制符应用，读取数据文件 tx.txt，建立数据集 txl。

```
%let   dir= 'd:\jx\tx.txt';     /*外部文件路径*/
filename   txsj   "(&dir)";        /*给路径起逻辑名字*/
data   txl;
        infile   txsj ;    /*读文件*/
        input     @1     qh       $4.      /*列指针控制读取*/
                  @5     tx_address     $17.
```

```
                @22    youbian       $4.
          ;
run;
```

【程序解读】 对外部文件的读取通过 INPUT 语句定义数据的开始位置和变量，@1 表示变量 qh 从第一列开始读取数据；$4.指明变量 qh 为字符类型，长度为 4，读取 4 个字符就结束。其他变量定义形式与变量 qh 类似。

（2）列控制符号模式

语法格式：n1−n2。

【语法解读】

n1：列开始位置，正整数值。

n2：列结束位置，正整数值。

【例 2.40】 对例 2.39 进行改造，通过列控制符号应用，读取数据文件 tx.txt，建立数据集 txsj。

```
%let     dir= 'd:\jx\tx.txt';     /*外部文件路径*/
filename   wbsj    "(&lj)";       /*给路径起逻辑名字*/
data    txsj;
            infile     wbsj ;   /*读文件*/
            input      qh        $ 1-4    /*列控制符号读取*/
                       tx_address      $ 5-21
                       youbian   $ 22-25
      ;
run;
```

【程序解读】 INPUT 语句中定义的变量 qh 的数据类型为字符类型，1−4 指明此变量对应的数据从数据文件中的第一列开始，到第四列结束。其他变量定义形式和定义变量 qh 类似。

【提示】 对同一问题通过不同的方法解决，达到同样的结果。对相同数据文件通过两种不同的列控制符号读取数据，程序运行后结果相同，具体选择哪种方式根据个人的喜好。

（3）单个@符号应用

@：INPUT 语句中单个“@”行控制符号，控制 SAS 行位置的指针，让指针控制在当前行，当遇到下个 INPUT 时行指针才移动。

【提示】 对数据块或数据文件进行过滤时经常用到该模式。

【例 2.41】 单个@符号应用，读取 d:\jx\credtype 数据文件，取出 B 类客户数据，建立数据集 btype。

```
    %let     dir= 'd:\jx\credtype.txt';
filename     wbsj    "(&dir)";
libname    csj     'd:\jx';
data    csj.btype;
            infile     wbsj;
            input    @20     card_type    $1.
                     @;          /*行控制符号,使指针控制在当前行，接着执行 if 条件判断语句*/
```

```
            if   card_type='B'   then do;     /*取 card_type='B'类型的卡的条件*/
                 input      @1    credt_bh        $3.
                          @4    card_num    $16.
                            ;
                             output    csj.btype;
                             end;
run;
proc    print    data=csj.btype    noobs;    /*打印输出 btyep 数据集，输出窗口不显示标号*/
run;
```

【程序解读】 语句 input @20 card_type $1. @;先读取 card_type 变量的数据，然后通过 if card_type ='B'语句判断，符合条件的就执行 DO END 语句块中 input 语句中定义的变量。

输出窗口显示输出结果，如图 2-9 所示。

输出 – （无标题）

SAS 系统　　　2012年04月10日

card_type	credt_bh	card_num
B	202	7345678987654322
B	305	6976543219876547

图 2-9　读取 B 类客户信息输出显示窗口

（4）@@符号控制读取

INPUT 语句中在读取外部数据文件或数据块中的数据时，若多条观测记录写在一行，为控制列中每个列对应数据到相应的列变量，需要通过“@@”行控制符起到按记录条数分割数据的作用，并按对应列读取数据。

【例 2.42】 @@符号读取控制应用，根据客户信息数据建立数据集 cust_inf。

```
data     cust_inf;
    length    address    $20.;/*长度超过 8 个字节，需要 length 语句先定义变量*/
                input     name   $   address   @@;   /*@@行控制符号读取多条记录在一行的记录*/
                 label     name='姓名'      address='通信地址';
        cards;
     高峰  北京市丰台区  章玉华  河北唐山  董小青  江苏南京
        ;
        RUN;
        Proc    print    data=cust_inf    label;    /*加 label 显示标签信息到输出窗口*/
         var    name    address;     /*改变输出显示顺序*/
run;
```

【程序解读】 cards 语句中的数据写在了同一行，没有按列对齐书写，通过@@行控制指针按 input 语句指定的变量对应读取数据。

输出窗口显示输出结果，如图 2-10 所示。

输出 － （无标题）

SAS 系统　　　　2012年04月10日

Obs	姓名	通信地址
1	高峰	北京市丰台区
2	章玉华	河北唐山
3	董小青	江苏南京

图 2-10　行控制符“@@”信息输出显示窗口

2.2.5　INPUT语句与PUT语句

INPUT 语句定义变量并读入数据，PUT 语句定义变量并读出数据，对比可以发现两者的区别。

1．INPUT语句

SAS 语言中数据步的建立离不开 INPUT 语句，在读入外部数据文件或 CARDS 语句后面的数据块时，通过 INPUT 语句定义变量、变量长度以及变量类型，并把对应的数据赋值给所定义的变量。

INPUT 语句的作用如下：

1）读取外部数据文件或 CARDS 语句后面的数据块。

2）定义变量、指定变量长度以及定义变量类型。

语法格式：INPUT <变量名 1 <$>w.d>　<变量名 2 <$>w.d>…;

【语法解读】

INPUT：定义变量关键字。INPUT 语句中定义变量，定义变量的对应类型，并可以指定读取的列格式。

变量名：用户定义的变量名。注意，命名要符合 SAS 命名规则。

$：$可选项，定义的变量为字符型的就加“$”符号，告诉 SAS 该变量为字符类型。

w.d：w 定义变量的总长度，整数型；如果数值型变量有小数，d 指明变量的小数位数。

【注意】 变量之间用空格分隔。

INPUT 语句中对变量定义有四种定义方式。

1）INPUT 语句中只是列出变量名，变量之间以空格分隔。对于字符类型的变量加一个“$”符号，字符类型长度取默认长度 8 个字符。数值类型的只写出变量名。

例如：INPUT　name　$　cred_num　;

优点：简单。

缺点：此类定义方式不能输入带空格的数据。输入的数据严格按变量对应的数据输入，字符类型的数据最多 8 个字符。

2）INPUT 语句中对定义变量用列格式定义对应的数据，变量名后用正整数指明变量对应的数据在数据行对应的列开始位置与结束位置。字符类型的变量加“$”符号。

例如：INPUT　car_id　$1-18　cred_num　21-36　credit_jf　19-20;

优点：此定义方式只需要标明变量对应的数据列的开始位置和结束位置就可以使变量读取到对应的数据；字符类型的变量可以包含空格；字符类型的数据可以读取到最大 200

个字符。

缺点：变量对应的数据输入时要严格按照列格式对应列输入，写程序时需要数出每一个变量对应列数据的开始位置和结束位置。

3）INPUT 语句中对定义的变量名后面直接定义输入类型的格式，定义变量的长度和数据类型。

例如：INPUT address $16. ed 3.;

优点：输入的数据不需要空格分隔。

缺点：输入的数据要严格按列对齐。

4）INPUT 语句中对定义的变量通过绝对指针方式读入输入变量对应的数据：@n 变量名 数据类型。

例如：INPUT @3 name $3.
@8 score 4.
;

优点：可以控制变量对应数据的开始位置，控制灵活。

缺点：需要把变量对应数据列的开始位置计算出来。

为帮助读者理解 INPUT 语句定义变量的方式，以及实际开发中如何根据数据书写格式选择 INPUT 语句的定义变量方式，通过案例分别讲解。

（1）列变量名方式读取数据

【例 2.43】 INPUT 语句中直接列出变量名和客户数据，建立数据集 credit。

```
  *input 语句后面定义变量，列出变量名和变量类型;
data     credit;
input    id   $    name   $    pj_score 5.1 ;    /*input 语句后面定义变量和变量类型*/
cards;
 1001  高洪伟  345.1
 1002  张东风  187.0
 1003  杨小飞  984.3
 ;
run;
```

【程序解读】 数据步 INPUT 语句中直接定义变量 id、name 和 pj_score，数据块按变量顺序赋值给变量。

（2）对变量名用列格式定义

【例 2.44】 读取外部数据文件 d:\jx\credtype.txt。

```
  *对变量名用列格式读取;
%let     dir= 'd:\jx\credtype.txt';
filename     wbsj    "(&dir)";
libname    csj     'd:\jx';
data    csj.lgsdq;
            infile     wbsj;
            input       credt_bh    $1-3
                    card_num    $4-19
```

```
                    card_type   $20
                    ;
run;
```

【程序解读】 INPUT 语句中定义的变量指定了列的开始位置和结束位置，严格按照数据对应列读取。

（3）变量名后面直接定义输入类型的格式

【例 2.45】 对例 2.44 进行改造，直接定义变量名类型的格式读入外部数据文件，建立数据集。

```
 *变量名后直接定义输入类型的格式;
%let     dir= 'd:\jx\credtype.txt';
filename    wbsj    "(&dir)";
libname   csj     'd:\jx';
data   csj.lgs;
            infile      wbsj;
            input       credt_bh    $3.
                    card_num    $16.
                    card_type     $1.
                    ;
run;
```

【程序解读】 INPUT 语句中定义的变量直接定义变量类型和长度，控制读取数据。

（4）绝对指针方式读入输入变量

【例 2.46】 对例 2.44 进行改造，用绝对指针方式读取外部数据文件，建立数据集。

```
 *绝对指针方式读入输入变量;
%let     dir= 'd:\jx\credtype.txt';
filename    wbsj    "(&dir)";
libname   csj     'd:\jx';
data   csj.gs;
            infile     wbsj;
            input   @1      credt_bh    $3.
                    @4 card_num    $16.
                    @20 card_type    $1.
                    ;
run;
```

【程序解读】 INPUT 语句中通过@指针定义每一个变量的开始位置，并定义变量类型和长度。

2．PUT语句

PUT 语句主要用来输出变量信息显示到指定的地方：

1）输出到 SAS 系统的日志窗口。

2）输出到 FILE 语句指定的外部文件。

3）输出到 SAS 系统的 OUTPUT 窗口。

【提示】 PUT 语句后面的定义和 INPUT 语句的定义变量语句相似。

语法格式：PUT <变量名 1 <$>w.d>　<变量名 2 <$>w.d> <_ODS_> <@|@@>;

【语法解读】

1）PUT：输出控制语句关键字。PUT 语句中定义变量或直接写变量名，定义变量对应的类型，并可以指定输出的列格式。

2）<_ODS_>：可选择项，对应用 file print ods 声明的变量指定此选项。

3）<@|@@>：控制输出指针，可以控制换行。

下面通过几个案例讲解 PUT 语句的作用。

【例 2.47】 PUT 语句输出数据集 gs 具有的变量 credt_bh 对应数据到日志窗口显示。

```
%let    dir= 'd:\jx\credtype.txt';
filename    wbsj   "(&dir)";
libname   csj    'd:\jx';
data   csj.gs;
            infile    wbsj;
            input   @1      credt_bh   $3.
                    @4    card_num   $16.
                    @20   card_type   $1.
                       ;
run;
*PUT 语句输出变量信息到日志窗口;
data   _null_;/*数据步只是做数据处理，数据集命名为_null_,这是特殊的数据集名字*/
set   csj.gs;/*读取数据集*/
put   credt_bh;/*输出变量 credt_bh 的数据到日志窗口显示*/
run;
```

【程序解读】 PUT 语句中直接写数据变量名字，把变量对应的数据显示到日志窗口。

【例 2.48】 对例 2.47 进行改造，PUT 语句输出数据集 gs 具有的变量 credt_bh 对应的数据到指定的外部文件。

```
*绝对指针方式读入输入变量;
%let    dir= 'd:\jx\filewrite.txt';
filename    wbsj   "(&dir)";
libname   csj    'd:\jx';
data   csj.gs;
            infile    wbsj;
            input   @1      credt_bh    $3.
                    @4    card_num   $16.
                    @20   card_type   $1.
                       ;
run;
*输出数据集数据到外部文件;
data   _null_;
file   wbsj; /*file 语句指定要写入的外部数据文件，一般与 put 语句一起使用*/
set   csj.gs;/*读取数据集*/
```

```
put @1 credt_bh $3. /*输出变量数据到外部文件*/
    @4 '|'
     @5 card_num $16.
     @21 '|'
     @22 card_type $1.
   ;
run;
```

【程序解读】 PUT 语句中直接写数据集变量名字，把变量对应的数据输出到外部文件，通过@指针控制数据输出到文件的列开始位置。

2.2.6 INFILE语句与FILE语句

1．INFILE语句

INFILE 语句是读入外部数据文件的关键语句。实际开发中大多数数据都是以外部数据文件的形式存储数据的。数据步的建立依靠的是数据，数据的获取需要 INFILE 语句告诉 SAS 系统外部数据文件存储在什么位置。INFILE 语句有时和 FILENAME 语句配合使用。FILENAME 用来指定文件路径所对应的逻辑名，物理位置是后面的文件路径。

INFILE 读取外部文件的设置

语法格式： INFILE ‘物理逻辑或逻辑文件名’<type> <options>;

【语法解读】

物理逻辑或逻辑位置名：指明物理文件路径或 filename 语句定义的逻辑文件名。

type：指明文件类型，默认是一个标准类型的外部文件。非标准文件类型要在这里指定，可以取的类型有 DLI、HFS、PIPE、IDMS、ISAM、VSAM、VTOC 等。这几个类型各自对应不同的操作系统。HFS 对应 UNIX 操作系统。MVS 对应 z/OS 操作系统。

options：指明主机中对应外部文件的选项，文件读取时经常用到的标准选项。

对于文件中每条记录长度不定，对应列数据有缺失值的文件，根据具体文件具体分析。对下面几个参数进行设置，来控制文件读取。

FLOWOVER：FLOW、OVER 这两个单词的组合。FLOW 这个单词翻译成中文是流动，也就是一直流动下去，直到数据文件结束。这是 SAS 的默认选项，如果读取的文件中不指明选项，默认的就是 FLOWOVER。它告诉 SAS 在读取数据外部文件时如果当前数据行记录长度和 INPUT 语句所声明的列对应数据长度不够，SAS 系统就从下一行记录读取数据，来满足列对应的数据。SAS 是先把数据读到缓冲区。

MISSOVER：该选项是对缺失数据列的数据处理。SAS 系统在读取数据行时，如果当前数据行长度到当前列没有达到 INPUT 语句声明的列长，SAS 系统就通过这个参数设置来告诉 SAS 系统，从当前列开始之后所有的列为缺失值。缺失值 SAS 默认用点（.）表示。不从下一行读取。

TRUNCOVER：TRUNCATE 和 OVER 的组合。TRUNCATE 翻译成中文是截取，从英文单词就可以知道此参数设置是告诉 SAS 系统在读取数据文件时，如果对应列当前读到的数据行长度不够 INPUT 语句声明中定义的列长，就把此列当前对应的数据截取出来给此列。如果读取到的数据行是最后一条记录，且当前列也没有数据值，那么从此列和此列之后的都赋值

给缺失值，用点（.）表示。

STOPOVER：STOP 和 OVER 这两个单词的组合，翻译成中文就是停止结束。如果在数据文件读取时 INFILE 里面设置了这个选项，就是告诉 SAS 系统在读取数据文件时如果读取到的当前数据行的长度不能满足 INPUT 语句中声明的列长度，SAS 就停止读取后面的数据，直接以当前已经读取的符合 INPUT 语句中定义列标准的数据建立数据集。

LRECL=：正整数的值，告诉 SAS 系统中的 INFILE 所读取的外部文件的逻辑记录长度，它依靠记录格式（RECFM=VALUE）来定，默认是 256 个字节。文件记录长度超过这个长度要在 LRECL 指定文件长度，最大是 32767。

RECFM=：指定文件数据格式（RECORD-FORMAT，RECFM），取值从下面指定的值中选择对应文件匹配的格式。

- F：数据文件数据块不固定，每条记录值的长度固定。
- V：数据文件数据块不固定，记录变长。
- FB：数据文件中数据块固定，每条记录值的长度固定。
- VB：数据文件中数据块固定，记录变长。
- U：数据文件中数据块不固定，记录变长。

FIRSTOBS 与 OBS 参数：

FIRSTOBS：FIRST 和 OBSERVATION 这两个单词的组合，告诉 SAS 系统当前读取的数据文件或数据块从哪条数据行记录开始读取数据文件或数据块，取值为正整数。

OBS：OBSERVATIONS 的缩写，翻译成中文是观察记录的意思。通过这个参数的设置告诉 SAS 系统读取当前数据文件或数据块到第几行记录结束。

【例 2.49】 读取客户基本信息数据文件，数据文件物理位置存储在“d:\jx\jxxx.txt”，建立客户信息数据集 cust_inf。

```
%let    dir= D:\jx\jxxx.txt;
%let   filename= "&dir";
data   cust_inf;
       Infile    &filename    missover    lrecl=270 ;
/*读取文件用了 missover 参数，文件记录长度超过默认 256,要用 LRECL 指定长度*/
                    input        id   $
                              name   $
                              sex   $
                              huf   $
                              address   $
                              zone   $
                              phone   $
                              myhf   3.
                              ysr   $
                              yj   $
                              qrrq   $
                              car   $
                              house   $
                              gjj   $
```

```
                    xyd    $
                    bankc    $
                    like1    $
                    like2    $
                    child    $
                    dw    $
                    xz    $
                    dwyb    $
                    zfamt    $
                    ylamt    $
                    yy    $
                    fybx    $
                    bx1    $
                    bx2    $
                    cust_bank    $
                    jxj    $
                ;
run;
```

【程序解读】 INFILE 中指定选项 missover 对缺失值处理，以点（.）表示，由于读取的外部文件长度超过默认长度 256，必须用选项 lrecl=指定。此文件中 lrecl=270 表示文件一行记录长度为 270 字节。

【提示】 通过此程序要学会根据文件的不同来判断用 INFILE 读取数据文件的哪些参数，具体参数设置要学会查 SAS 帮助。由于 INFILE 里面的参数太多，不能一一讲解。这个程序只是其中的一类，文件长度超过了默认文件长度 256，要用 LRECL=指定文件长度。如果不加这个参数则后面的会被截断。

【例 2.50】 取数据文件“d:\jx\zone.dat”的第一条至第 5 条数据，生成数据集 zone5。

```
%let    dir = D:\jx\zone.dat;
filename    wjlj    "&dir";/*定位外部文件逻辑路径名*/
data    zone5;
            Infile    wjlj    truncover    firstobs=1    obs=5 ;
/*读取文件用了 firsobs 和  obs 两个参数，来控制读取文件记录条数*/
input    dq          $
           youbian    $
        ;
run;
```

【说明】 INFILE 后面可以直接跟文件路径，但这不是一种很好的编程规范，不利于程序的修改和阅读。不提倡用此方式，建议 INFILE 语句与 FILENAME 语句联合应用。

2．FILE语句

SAS 语言写入外部数据文件是通过 FILE 语句实现其功能的。对比 INFILE 语句和 FILE 语句实现的功能，可以发现其不同之处，INFILE 语句和 FILE 语句可以说是一对，只是功能方面 INFILE 语句是读外部数据文件或数据块，FILE 语句是把数据写到指定目录的数据文件。FILE 语句是把 PUT 语句后面指定的变量所对应的数据输出到指定文件，要和 PUT 一起使用。

语法格式：FILE 文件名 <ENCODING='编码类型'> <选项> <主机选项> ;

【语法解读】

1）文件名：可以取的选项如下。

① 指定要写入外部文件的文件名，可以直接写完整路径下的文件名。

② 可以写宏变量名，宏变量指定外部文件的完整路径。

③ 可以写到 SAS 日志窗口，取 LOG 选项就可以。

④ 可以写到 SAS 的 OUTPUT 窗口，取 PRINT 选项就可以。

2）ENCODING='编码格式'>：可选项，指定要写入到的文件编码，告诉 SAS 写入到的文件是什么编码。

3）<选项>：可选项。写入到文件的控制的选项比较多，常用文件选项如下：

① BLKSIZE=指定写入到的文件大小。

② LRECL=正整数的值，告诉 SAS 系统中的 INFILE 所写入的外部文件的逻辑记录长度，它依靠记录格式（RECFM=VALUE）来定，默认是 256 个字节。文件记录长度超过这个长度要在 LRECL 指定文件长度，最大是 32767。

③ RECFM=文件的数据格式（RECORD-FORMAT，RECFM），取值从下面指定的值中选择对应文件匹配的格式。

- F：数据文件数据块不固定，每条记录值的长度固定。
- V：数据文件数据块不固定，记录变长。
- FB：数据文件中数据块固定，每条记录值的长度固定。
- VB：数据文件中数据块固定，记录变长。
- U：数据文件中数据块不固定，记录变长。

④ DROPOVER：DROP 和 OVER 这两个单词的组合。通过单词的意思就可以知道，如果写入的文件超出规定的数据行的长度，就丢失它。

⑤ FLOWOVER：SAS 的默认选项，如果写入的文件中不指明选项，默认就是 FLOWOVER，它告诉 SAS 系统在写入到数据外部文件时如果当前数据行的记录长度超过规定的行记录长度，就把超出的行记录写到下一行。

⑥ DSD：Delimiter、Sensitive 和 Data 的缩写，翻译成中文是敏感数据分割。这个选项用来告诉 SAS 系统数据包含嵌入的符号，如制表符、逗号或用引号括起来的值。这样分隔符号就可以包含在数据中写入文件。指定 DSD 选项后默认分割符是逗号。

⑦ MOD：在写入的文件中任何行对输出之后就写输出行到输出。如果不加这个参数默认是 OLD 的输出。

4）<主机选项>：可选项，是在 UNIX 操作系统环境下的选项。但有几个选项在 Windows 操作系统环境下也可用（如 LRECL=和 RECFM=）。

① BLKSIZE=或写成 BLK=：指定写入文件的字节数，默认是 8KB，最大为 1GB。

② NEW|OLD：告诉 SAS 系统把打开的新的文件作为输出，如果文件名存在，则把这个存在文件删除并重建。这是个默认的选项动作。

③ UNBUF：告诉 SAS 系统不要把缓冲区数据写入到文件。此选项特别适合在编写的数据收集程序中限制缓冲区数据写入到文件。

为便于深刻理解和运用 FILE 语句，下面举几个实例。

【例 2.51】 FILE 语句应用，把数据集 cust 写入到 d:\jx\cust.txt 文件中，列之间用“|”分割数据。

程序如下：

```
data   cust;
Input   id   $   name   $   jf;
cards;
001  刘小红  300
002  张大发  5600
003  马微雨  345
;
run;
data   _NULL_;/*只是数据处理，不生成数据集，命名数据集为_null_*/
set    cust;/*set 语句读取 cust 数据集*/
file   'd:\jx\cust.txt';        /*file 语句，把数据集数据写入到指定目录的文件中*/
put   @1   id   $  /*put 语句输出列内容*/
         @4   '|'
         @5   name   $
           @11   '|'
           @12   jf   5.
         ;
run;
```

【注意】 一个中文字符占两个字节，要注意@n 标的是每个列的开始位置。用 UltraEdit 可以看到每个列的开始位置。输出分隔符占一列，要计算在内。

【例 2.52】 以 Oracle 数据库里的数据表 jx_inf 为例，把该表里的数据写入到 d:\jx\cust_inf.dat 目录下的文件里，以分隔符“|”分割数据列，目的是学习 LRECL=参数的应用。

实验步骤如下：

1）先在 Oracle 数据库里输下面的建表语句。

create table jx_inf (id char(8),host varchar2(15),address varchar2(30),fees number(5,1), time_len number(4),arrears_count number(3),last_mtimel timestamp,curr_mtimel timestamp,next_mtimel timestamp);

2）在 Oracle 数据库里执行插入语句到表中。

insert into jx_inf values ('12345676','刘小红','北京市丰台区丰台总部', 300, 20, 7, to_date ('2012-04-17 09:01:10','YYYY-MM-DD HH24:MI:SS'),to_date('2012-04-16 10:01:10', 'YYYY-MM-DD HH24:MI:SS'),to_date('2012-04-18 12:01:10', 'YYYY-MM-DD HH24:MI:SS'));

3）在 SAS 编辑窗口执行下面的 SAS 程序。

【提示】 登录数据库的 user=登录数据库的用户名， password=登录数据库的密码，

path=登录的数据库。要把 LIBNAME 语句的这 3 项改成你的数据库信息。

%LET out='d:\jx\cust_inf.dat'，这里的路径 d:\jx 要先建好，cust_inf.dat 不用建，执行后会生成此文件。

程序如下：

```
libname  jx  oracle  user=chiran  password=chiran  path=orcl;
     /*建立与 Oracle 数据库连接的逻辑库*/
%let  out='d:\jx\cust_inf.dat';  /*建立指向目录的文件路径的宏变量*/
data   _NULL_;
file  &out  lrecl=134; /*FILE 语句把数据写入到指定目录的文件中*/
set  jx.jx_inf;  /*读数据库中的表 jx_inf*/
put   @1  id    $8./*PUT 语句把表数据输出到外部文件*/
       @9  '|'
        @10  host  $15.
        @25  '|'
        @26 address  $30.
        @56  '|'
        @57  fees  5.1
        @62  '|'
        @63  time_len  4.
        @67  '|'
        @68  arrears_count  3.
       @71  '|'
       @72  last_mtimel  datetime20.1
       @92  '|'
       @93  curr_mtimel  datetime20.1
       @113  '|'
       @114  next_mtimel datetime20.1
       @134  '|'
     ;
run;
```

【程序解读】 file 语句中指定 LRECL=选项，告诉 SAS 系统一行记录的长度是多少。

【说明】 FILE 在写入到外部文件时如果行记录长度超过 256，要用 LRECL=参数指定；如果不超过这个长度可以省略此参数。

2.2.7 DELIMITER语句与LENGTH语句

1．DELIMITER语句

DELIMITER 语句主要用来告诉 SAS 系统变量之间的数据以什么类型分割，中文意思为定界符、分隔符。通过这个单词可以知道 SAS 用这个语法来处理数据文件或数据块数据之间用分隔符分割的数据。SAS 系统默认数据文件或数据块数据之间以空格分割。如果所给数据文件或数据块数据之间是以空格分割的，就可以省略这个参数。通过 DELIMITER 告诉 SAS 系统数据文件或数据块数据之间遇到分隔符为一个间隔列，可以把数据赋值给 INPUT 语句中定义的列变量。SAS 语法引用该语句时可以简写成 DLM。

语法格式：DELIMITER='分隔符'或简写为 DLM='分割符'

【例 2.53】 外部数据文件 bl.dat 以逗号“,”分隔符分割数据。数据文件存储位置为 d:\jx\bl.dat。数据文件格式为 dat 格式。请以此数据文件建立数据集 bl。

```
%let  dir= 'd:\jx\bl.dat';
```

```
filename    wbwj   "(&dir)";
data   bl;
  Infile   wbwj   dlm=',' ;     /*文件以逗号“,”分割，指定文件分隔符 dlm=','*/
    length     fh    $50.;
          input   bh $    fh     jyl    cs    sr;
run;
```

【程序解读】 INFILE 语句中，读取外部数据文件之间以逗号分割的变量数据通过 dlm=',' 告诉 SAS 系统以分隔符号表示变量对应数据结束。

【例 2.54】 数据步读入 CARDS 数据块的数据，数据分隔符号为“|”，建立数据集 cust_dlm。

程序如下：

```
  data    cust_dlm;
   infile    cards    dlm='|';    /*读入 cards 数据块中以 dlm='|'分割的数据*/
     input    qh    $    mobile    zone    $;
cards;
101|62378854341|010|
102|12376598763|031|
;
run;
```

【提示】 如果分隔符是 Windows 操作系统的制表符号，单引号外要加 x。如 DLM='09'x。这个 x 告诉 SAS 系统是 Windows 操作系统。如果是其他操作系统就查系统标识，具体问题具体分析。常用的数据文件或数据块数据之间的分隔符如下：“|”、“！”、“,”。

2．LENGTH语句

SAS 语言对定义的变量有默认值，默认变量的长度为 8 个字节，超过 8 个字节长度的变量，要首先用 LENGTH 语句来定义变量，并指定长度。

语法格式：LENGTH 变量名 <$> 变量长度;

【语法解读】 LENGTH 是关键字，如果变量是字符类型的，定义变量类型前面加“$”符号。

【提示】 LENGTH 语句与 INPUT 语句一起使用时必须把 LENGTH 语句放在 INPUT 语句前面，INPUT 语句中引用 LENGTH 语句中定义的变量不需要再定义，直接引用变量名就可以了。

【例 2.55】 LENGTH 语句应用，地址变量数据超过默认长度，建立数据集 cust_addr。

程序如下：

```
data    cust_addr;
length    address    $30.;
/*length 语句定义变量 address，指定数据类型长度为字符类型，长度为 30*/
input    id    $    name    $    sex    $    address;
cards;
  1001  高小名 F 北京市西城区广安门外大街 63 号院 9 号楼 1 单元
  1002  李小雷 M 山东省济南市槐荫区
  ;
run;
```

【程序解读】 由于变量 address 的字符长度超过了默认长度 8 个字节，需要通过 LENGTH 语句定义变量，并定义变量类型。INPUT 语句中直接引用此变量名，不需要再定义变量类型。

2.2.8 日期与时间定义

SAS 语言中的数据类型有两种，分别是字符型和数值型。SAS 中日期与时间的处理选择了把日期和时间转换成数值型来存储。SAS 人为地约定 1960 年 1 月 1 日这一天为开始日期，作为基准日期，把此日期定义为零（0）这个数值类型存储在 SAS 中，其他日期的存储是通过从 1960 年 1 月 1 日到所给日期计算出的天数间隔值来存储的。1960 年 1 月 1 日之前的存储为负值。相当于日期之间做减法运算，来求天数。时间的存储值也是从零（0）时开始，以秒为单位进行计算。

当前日期为 yyyy 年 m 月 n 日

天数的计算公式=当前年份（yyyy）的天数+（yyyy−1）年的天数+…+1961 年的天数+当前 1 月份天数+当前 2 月份天数+…+（m−1）月份的天数+（n−1）的天数

SAS 在日期的读入时有自己定义的日期和时间格式，在对外输出时也有自己定义的输出日期和时间格式，这是 SAS 的特殊之处。下面举例来进一步讲解 SAS 日期和时间，以及它们的应用。

1．读入日期的格式

（1）DATEw.

【说明】 DATE 是日期关键字，w 指定日期长度。默认是 7，超过 7 的长度要定义。读入的日期书写格式为 ddmmmyy 或 ddmmmyyyy 的格式日期（dd 指的是所在月的某一天，取值为 01 到 31 的某一天，mmm 指用英文书写月份的前 3 个字母，yyyy 指 4 位表示的年份）。

【例 2.56】 读入客户开卡信息，建立数据集 opencard。

```
data    open_card;
Input   id_nm   $   name   $   open_dt   date.;   /*open_dt 用的默认日期*/
cards;
 10001  高小红  10mar12
 10002  刘晓霞  18apr12
;
run;
proc   print   label;
format   open_dt   yymmdd10.;/*format 语句定义了日期输出显示格式*/
run;
```

输出窗口显示输出结果，如图 2-11 所示。

```
输出 - (无标题)
                SAS 系统            2012年04月12

   Obs    id_nm    name      open_dt

    1     10001   高小红   2012-03-10
    2     10002   刘晓霞   2012-04-18
```

图 2-11　客户开卡信息输出窗口

（2）DATETIMEw.d

【语法解读】 DATETIME 是定义读入日期时间格式的关键字，w 指定日期时间长度，w 值要取偶数，默认是 16，d 用来指定秒的值。注意 d 的值要比 w 的值小。

【例 2.57】 交易时间到秒，建立数据集 trandt。

程序如下：

```
  data   trandt;
    length     card_nm   $16.;
    input   card_nm     trade_time     datetime 18.1;
/*datetime18.1 定义读入 trade_time 用的日期时间*/
label   card_nm='卡号'     trade_time='交易时间';
cards;
1603134567812323 13apr12:9:28:56.8
2000197645231586 12apr12:10:30:36.7
;
run;
```

（3）JULIANw.

【语法解读】 JULIAN 是日期关键字，w 是定义的日期长度，默认是 5。读入日期书写格式为 yydd 或 yyyyddd。yy 和 yyyy 分别代表读入的两位和 4 位的年份。ddd 是从当前年初到当前年份日期算起的天数，范围为从 01～365 的值。

2．输出日期的格式

SAS 系统在对输出日期格式方面有更多输出日期格式的设置，和上面讲的读入日期时间的格式一样。输出日期格式有很多，常用的输出日期格式有：DATEw.、DATETIMEw.d、DAYw.、MONTHw.、YEARw.等。其他日期可以查 SAS 帮助文档。

2.2.9 INFORMAT与FORMAT定义数据格式

1．INFORMAT语句

实际开发中SAS系统默认变量属性的设置满足不了需求，因此SAS语言通过INFORMAT语句根据实际需求定义变量输入格式。

语法格式：INFORAMT 变量名 数据类型格式;

【例 2.58】 INFORMAT 语句应用，建立数据集 in_sy。

程序如下：

```
  data   in_sy;
   informat     sy_dt     mmddyy10.;
/*根据输入数据时间，定义读入变量 sy_dt 的数据日期格式*/
   length   yj   $30.   ;
   input     yj $      sy_dt;
cards;
  北京   10-23-2010
  山东   09-08-2011
  ;
run;
```

```
proc    print;
   format    sy_dt      yymmdd10.;
run;
```

【程序解读】 INFORMAT 语句定义变量 sy_dt 并指定输入数据类型格式为 mmddyy10.。

2．FORMAT语句

FORAMT 语句的设置和 INFORMAT 语句的设置基本一样。不同点是 FORMAT 语句定义变量的输出数据格式，与 INFORMAT 相反，但定义方式一样。

语法格式：FORMAT 变量名 定义数据类型格式。

【例 2.59】 FORMAT 格式应用，对例 2.58 进行改造。

程序如下：

```
data    in_syformat;
   informat     sy_dt      mmddyy10.;
/*根据输入数据时间，定义读入变量 sy_dt 的数据日期格式*/

   format    sy_dt      yymmdd10.;
/*根据数据时间，定义变量输出 sy_dt 的数据日期格式*/

  length    yj    $30.    ;
  input    yj $        sy_dt;
cards;
  北京    10-23-2010
  山东    09-08-2011
  ;
run;
```

【程序解读】 FORMAT 格式指定变量 sy_dt 的输出格式为 yymmdd10.。

【提示】 FORAMT 在数据步和 PRINT 过程步中都可以应用。

2.2.10 LABEL 语句与 RETAIN 语句

1．LABEL语句

实际项目开发中经常需要为变量加入描述信息，以便其他编程人员理解此变量代表什么。SAS 语言中用 LABEL 语句给变量定义描述信息标识。描述信息和变量名一起被存储到数据集中。

语法格式：LABEL 变量名='变量描述信息';

【语法解读】

LABEL：定义变量标签的关键字。

变量描述信息：最多 40 个字符。

【例 2.60】 LABEL 语句应用，根据学生信息，建立数据集 students。

程序如下：

```
data    students;
 Length    address    $40.;
 input    id    $    name    $    sex    $    address    $ ;
```

```
  Label   id='学号'   name='姓名'   sex='性别'   address='家庭住址' ;
  /*label 语句对变量添加描述*/
cards;
 1001 刘晓燕 F 北京市西城区广安门外大街
 1002 杨小光 M 北京市东城区 60 号院 2 号楼 1 单元 603 室
 ;
 run;
 proc print   label; /*对打印输出的信息加 label 选项才能打印出标签的定义描述*/
 run;
```

【程序解读】 LABEL 语句对 INPUT 语句中定义的变量指定描述性信息，便于用户理解每个变量代表什么意义。

【注意】 调用 PRINT 过程时打印标签信息必须加 LABEL 选项。

2．RETAIN语句

为提高程序效率，使变量具有的数据信息保留在内存中，直到退出当前程序才失效，SAS 语言通过 RETAIN 语句对需要赋值的变量进行声明。

语法格式：RETAIN 变量名 初始值;

【语法解读】

RETAIN：关键字。

变量名：赋值的变量。

初始值：具体值。

【例 2.61】 RETAIN 语句应用，求 y 的三次幂，x 的初始值为 2。

程序如下：

```
 *retain 语句赋初始值给变量 x;
data   mul;
 retain   x   2;   /*x 初始值赋值为 2*/
   y=x**3;/*y 值为 x 的三次幂*/
run;
```

2.2.11 RENAME语句与数组语句

1．RENAME语句

实际项目开发中对数据二次处理时经常需要改变变量的名字，以便理解或与其他数据集的变量名统一。SAS 语言对数据集中的变量或 INPUT 语句中定义的变量通过 RENAME 语句修改变量名。

语法格式：RENAME =(旧变量名=新变量名);

【语法解读】 RENAME 是关键字。

功能：修改变量名字。

【例 2.62】 RENAME 语句应用，将数据集中的变量名 id 修改为 id_nm。

程序如下：

```
 data   rename_id   (rename=(id=id_nm));   /*变量 id 修改为 id_nm*/
```

```
        input    id   name   $   sex   $   jf;
    cards;
    1200031  刘小光  M 3000
    1100089  马小美  F 9000
;
run;
```

2．数组语句

数组是在程序设计中，为了处理方便把具有相同类型的若干变量按有序的形式组织起来的一种形式。这些按序排列的同类数据元素的集合称为数组。

SAS 中数组存储的是变量，不是数据，数组只在数据步（DATA STEP）有效，数组的标号从“1”开始。

语法格式：ARRAY 数组名 {m<,n>} <$> <_temporary_> <数组变量列表>;

【语法解读】

ARRAY：定义数组的关键字。

数组名：给数组起的名字。

{m<,n>}：m 数据元素的个数，<,n>只有在二维数组时才有，可选项。

<$>：可选项，数组为字符类型时才有此项。

<数组变量列表>：可选项，变量名的列表。

【注意】 三维和二维定义方式一样。

【例 2.63】 定义一个一维数组，存储学生姓名。

```
data    stu_name;
  array   name{3}   $   ('张小红' '刘晓峰'   '上官晓红');    /*定义一个字符类型的一维数组*/
run;
```

【例 2.64】 定义一个二维临时数组。

```
data   twotemarr;
  array   temp{2,2} $    _temporary_ ('A' 'B' 'C' 'D');
  /*定义二维字符类型临时数组*/
  y=temp(2,2);
  put   y=;
run;
```

日志窗口显示 y=D。

【提示】 通过上面可以看出临时数据不保存数据集，在数据集里为空，只是作临时处理数据用。

2.2.12 SAS编程注释与OPTIONS语句

1．SAS编程注释

书写程序养成添加注释的习惯是为了增强程序的可读性，让其他人更好地理解程序的功能和用途。

SAS 添加注释的方式有两种，根据实际开发的需要可以选择合适的方式。

1）第一种方式：*书写注释语句;。

【说明】 以*（星号）开始，中间写注释语句，以“;”分号结束。这里的分号是在输入法为英文的状态下输入的分号，和程序中用到的分号一样。

2）第二种方式：/* 注释语句 */。

【说明】 以“/*”开始，中间写注释语句，以“*/”结束。

2．OPTIONS语句

OPTIONS 语句主要用来临时改变用户安装 SAS 时的 SAS 系统选项中的选项设置。通过 OPTIONS 语句改变的 SAS 会话或作业中的设置会一直有效，直到再次通过 OPTIONS 语句改变。OPTIONS 语句可以出现在 SAS 系统的任何地方。若 OPTIONS 语句放在数据步内或过程步内，此语句只对该步有效；若放在某一步之外，此语句只对跟随在 OPTIONS 语句后的那一步有效。

OPTIONS 语句语法格式：OPTIONS 选项。

【语法解读】 OPTIONS：关键字。

选项：改变 SAS 系统默认选项设置的选项。

OPTIONS 具有的常用选项及功能说明如表 2-4 所示。

表 2-4　OPTIONS 具有的常用选项及功能说明

选　项	功 能 说 明
date	输出页显示日期，SAS 系统默认输出页显示日期
nodate	输出页不显示日期
center	居中设置：输出页信息居中，SAS 系统默认输出信息居中
nocenter	输出页信息不居中
number	页码设置：有页码
nonumber	输出无页码
linsize=	行宽设置
pagesize=	指定每个输出页显示的行数，取值范围为[15，32767]的整数值
firstobs=|obs=	对显示数据集设置：指定从第几条记录开始到第几条记录结束
mprint	在日志中对宏语句显示信息设置：打印输出宏语句信息
nomprint	不显示宏语句信息到日志
mlogic	对宏过程执行进行跟踪：对宏过程执行情况写日志到日志窗口
nomlogic	不写宏过程执行情况到日志窗口
missing=	指定数据集中数值型的缺失值用什么字符类型替换
syntaxcheck	对程序语法检查设置：需要语法检查
nosyntaxcheck	不需要语法检查
dmssynchk	SAS Windows 环境中启动对多个步的语法检查：启动语法检查
nodmssynchk	不启动语法检查
notes	日志窗口默认显示注释
nonotes	设置日志窗口不显示注释
nosource	日志窗口不显示编写的程序信息

【例 2.65】 通过 OPTIONS 语句改变数据集默认设置。通过宏查询数据集中符合条件的数据信息。

```
%macro   define_op(v_bh); /*宏过程开始*/
options   obs=2   compress = yes ;
/*options 语句中 obs=3 指明取数据集的前二条数据，compress=yes 指明对数据集压缩*/
options   mlogic   mprint   nosyntaxcheck   nodmssynchk;
/*options 语句中 mlogic 指明对宏过程跟踪写日志到日志窗口，mprint 指明打印输出宏信息*/
/*nosyntaxcheck 指明不需要语法检查，nodmssynchk 对启动的多个 SAS 步程序不启动语法检查*/
/*数据处理*/
%let   dir= 'd:\jx\bl.dat';
Filename    wbwj    "(&dir)";
data   blsj;
infile   wbwj   dlm=',' ;/*文件以逗号“,”分割，指定文件分隔符 dlm=’,’*/
length    fh   $50.;
        input   bh $   fh    jyl   cs   sr;
run;
/*通过 SQL 过程实现根据条件查询数据集信息*/
proc   sql;
select   * from   blsj   where   bh=&v_bh;   /*引用宏参数*/
quit;
%mend   define_op;    /*宏过程结束标志*/
%define_op('01002');   /*宏调用，并传递实参 01002 给宏变量 v_bh*/
```

【程序解读】 本程序通过宏过程引用了 OPTIONS 语句改变 SAS 系统的默认选项设置。具体宏过程后面章节会有详细的讲解，本案例重点学习 OPTIONS 语句的应用。

第 2 篇　提　高　篇

第 3 章　数据步基础与案例

3.1　数据步基础

SAS 系统分为两个步：数据步和过程步。数据步是 SAS 系统处理数据的核心部分，SAS 系统通过数据步对外部数据文件或其他数据库中的表数据或已经生成的数据集进行处理，生成 SAS 能识别的数据。数据步就是一个动态处理数据的过程，数据步处理完成后生成的数据存储到逻辑库对应的目录中，SAS 其他模块或过程步调用数据步生成的数据，数据步生成的数据集可以作为数据分析、数据挖掘和创建模型的数据基础。

3.1.1　数据步概述与定义

1．数据步概述

实际的开发中经常遇到的是来自其他系统的数据，如关系数据库数据（Oracle、DB2、Teredate、SASSPDS）、主机 COBOL 语言生成的数据文件、外部的其他文本文件数据等外部数据，对于这些数据 SAS 系统要对其能够识别和利用，首先需要通过 SAS 系统对这些外部数据进行处理，SAS 系统处理数据的过程称为数据步。因此归纳总结可以得出数据步主要是处理数据的过程。无论用 SAS 的哪个模块做应用研究和分析处理，首先都要有正确有效的数据，数据步处理数据的优劣直接影响到其他模块分析的正确性。数据步是 SAS 系统处理数据的核心步，学好 SAS 数据步是学习 SAS 其他模块的基础。

2．数据步定义

理解 SAS 系统数据步语法定义是学好数据步的关键，数据步以“DATA”关键字为开始标志，以“RUN”语句为结束标志，每个语句的结束符号为“;”（分号）。

数据步语法格式：DATA　<数据集名>　<参数选项>;

　　　　　　　　　　SAS 处理数据集语句;

　　　　　　　　RUN;

【语法解读】

DATA：数据步开始标志关键字。

数据集名：用户定义的数据集名，遵守 SAS 命名规范，省略此项为 SAS 默认数据集名。

参数选项：指定设置数据集的选项，如 KEEP、DROP、FIRSTOBS=、OBS=。

SAS 处理数据集语句：对数据集读入、复制、修改和合并、条件判断等基本处理数据语句。

RUN：数据步结束标志

数据步生成的就是 SAS 能识别的一个二维表数据，如果读者学过其他关系数据库，如 ORACLE、DB2 等，就能很容易地理解数据步生成的数据集。其实数据集名就是一个表的名字，只是叫法不同。

【例 3.1】 学生信息数据，数据集名字为 students，保留变量 name、sex、class、address。

```
data    students   (keep=name   sex   class   address);
/*data 数据步开始标志，数据集名为 students，keep=语句取要保留的字段*/
   length     address $30.;
   /*对应字符类型的变量其长度超过默认长度 8 个字符要通过 length 语句先定义变量*/
   input   id   name   $   sex   $   class   address;
   /*定义变量，其中 address 为 length 语句中定义好的变量*/
cards;   /*读 cards  后的数据*/
1001  高溪红  F 1  北京市西城区广外大街
1002  张明明  M 2  北京市东城区
;   /*数据块结束这里要有个分号*/
run;   /*数据步结束*/
```

【程序解读】

1）data：告诉 SAS 系统数据步开始。该关键字后面的 students 为数据集名字，后面括号里的参数选项 keep=取数据集要保留的字段。

2）address 变量长度超过默认字符类型长度 8 个字符，需要通过 length 语句定义变量，而不能在 input 语句中定义变量。

3）input 语句：定义变量，并引用 length 语句中定义的变量。

4）cards 语句：后面为要读入的数据，当数据结束时后面跟“;”(分号)。

5）run：告诉 SAS 系统到此处结束。

【提示】 实际应用开发中 if 语句与 where 语句是数据步经常用到的条件过滤语句。因此读者要重点学习 if 语句与 where 语句的实际开发应用，通过下面的案例深刻理解条件过滤语句的应用。

【例 3.2】 数据步过滤语句应用，if 条件语句应用。对客户数据 cust.dat 处理，提取数据文件中城市为北京和青岛的数据，生成数据集 open_bjqd，其他城市的客户数据生成到 other 数据集，存储到 d:\jx 目录中。

```
libname   jx   'd:\jx';                      /*定义数据集存储物理路径的逻辑库*/
   %let   dir= 'd:\jx\cust.dat';             /*外部数据文件的路径*/
   filename   fil   "(&dir)";                /*此语句指定定义逻辑文件名*/
data   jx.open_bjqd   jx.other;              /*同时建立两个数据集，分别为 open_bjqd 和 other*/
   infile   fil ;                            /*读入外部文件*/
   input   zone   $   open_count    city   $;
        if   city   in ('北京','青岛')   then   output   jx.open_bjqd;
  /*if 语句条件判断，output 语句后面为输出到的数据集名*/
/*把北京和青岛两大城市的客户数据存储到数据集 jx.open_bjqd*/
               else   output   jx.other;      /*把其他城市客户数据存储到数据集 jx.other*/
run;
```

【程序解读】

1）if city in ('北京','青岛') then output jx.open_bjqd; 此 if 语句判断 city 为 in 这个语句范围内的城市客户数据，输出到 jx.open_bjqd 数据集。

2）else output jx.other; 此 else 语句把不满足 if 语句的数据输出到 jx.other 数据集。

【提示】 if 条件语句中 then 语句后面必须加 output 语句才能输出到数据集。in 语句里为条件具有的值，此处 in ('北京','青岛')表示 city 变量的值如果为北京、青岛两个城市的数据就取出来输出到数据集 jx.open_bjqd。

【例 3.3】 数据步 where 条件过滤数据集。对例 3.2 进行改造，用 where 语句生成符合条件的数据集。

```
libname  jx  'd:\jx';   /*定义数据集存储物理路径的逻辑库*/
  %let  dir= 'd:\jx\cust.dat'; /*外部数据文件的路径*/
  filename  fil  "(&dir)";  /*此语句指定定义逻辑文件名*/
  *先对外部数据文件处理，生成数据集 cust;
data  cust;  /*生成数据集 cust*/
 infile  fil ;  /*读入外部文件*/
 input  zone  $  open_count  city  $;
run;
*对 cust 数据集二次处理，把北京和青岛两个城市的数据取出来生成数据集 bjqd;
data  jx.bjqd;  /*数据集 bjqd*/
  set  cust  (where=(city  in  ('北京','青岛')));  /*where 语句把 city 为北京和青岛的数据取出来*/
run;
*对 cust 数据集二次处理，把不是北京和青岛两个城市的数据取出来生成数据集 othercity*/
data   jx.othercity;  /*数据集 othercity*/
  set cust (where=(city not in  ('北京','青岛'))); /*where 语句把 city 不是北京和青岛的数据取出来*/
run;
```

【程序解读】

1）where=(city in ('北京','青岛'))语句取出 city 对应城市为北京和青岛的数据。

2）where=(city not in ('北京','青岛'))语句取出 city 对应城市不为北京和青岛的数据。

【提示】 对例 3.2 和例 3.3 对比学习，可以发现 where 语句效率比较高，where 语句是批量处理，先把符合条件的数据取出来放到内存，然后批量生成。而 if 语句是一条条读取，效率比较低。对于实际应用数据一般都在百万级以上，一定要考虑程序处理数据的效率。

实际的应用中数据步经常用在宏过程中，数据步与宏过程的联合应用才能体现强大的功能。

【例 3.4】 数据步与宏过程的联合应用，对例 3.2 进行改造，用宏过程实现取每一个城市的数据，生成对应的数据集。

```
libname  jx  'd:\jx';   /*定义数据集存储物理路径的逻辑库*/
%let  dir= 'd:\jx\cust.dat'; /*外部数据文件的路径*/
filename  fil  "(&dir)"; /*此语句指定定义逻辑文件名*/
*先对外部数据文件处理，生成数据集 cust;
data  cust;  /*生成数据集 cust*/
infile  fil ;  /*读入外部文件*/
```

```
    input   zone   $   open_count   city   $;
    run;
*对 cust 数据集二次处理，把北京和青岛两个城市的数据取出来生成数据集 bjqd;
%macro   city (v_city); /*定义宏，宏名为 city*/
data    jx.city;   /*数据集名 city*/
    set   cust   (where=(city="&v_city"));   /*city=宏变量对应的值*/
run;
%mend    city;
%city(北京);   /*调宏过程 city，传递实参北京给宏变量 v_city*/
```

【程序解读】

1）宏过程 city 中定义了宏变量 v_city，根据传递的参数取数据集符合条件的数据。

2）where=(city="&v_city")条件语句根据宏变量的值而取出不同的数据，灵活性更强。宏过程中引用宏变量用双引号，不能用单引号。

【例 3.5】 通过宏过程实现取出符合 zone 条件的数据。

```
libname   jx   'd:\jx';   /*定义数据集存储物理路径的逻辑库*/
%let   dir= 'd:\jx\cust.dat'; /*外部数据文件的路径*/
filename   fil   "(&dir)"; /*此语句指定定义逻辑文件名*/
*先对外部数据文件处理，生成数据集 cust;
data   cust;   /*生成数据集 cust*/
    infile   fil ;   /*读入外部文件*/
    input   zone   $   open_count   city   $;
run;
*对 cust 数据集二次处理，把北京和青岛两个城市的数据取出来生成数据集 bjqd;
%macro   zone (v_zone);   /*定义宏，宏名为 zone*/
data   jx.z_&v_zone;     /*数据集名 z_&v_zone，v_zone 的值数据集名是变化的*/
    set   cust   (where=(zone="&v_zone"));   /*zone=宏变量对应的值*/
run;
%mend   zone;
%zone(1002);
```

【程序解读】

data　jx.z_&v_zone;数据集名字为变量传递过来的命名组合，是变化的数据集名。

【例 3.6】 综合 SQL 过程、数据步和宏过程 3 部分，提取数据集 cust_inf 符合条件的数据。

```
libname   jxsj   'd:\jx';   /*建立数据存储位置的逻辑库*/
proc   sql   noprint;   /*调 SAS 内部的 SQL 过程，noprint 选项告诉 SAS 不打印输出，提高效率*/
/*创建表 SQL 语句*/
create   table   jxsj.cust_inf
(id   char(6)   label='客户号',   name   varchar(10)   label='客户姓名',   tran_dt   num
format=is8601dt20.   label='交易时间',
address   varchar(100)   label='交易地点');
quit;   /*SQL 过程的结束交互标志位 quit*/
proc   sql   noprint;
    insert   into   jxsj.cust_inf   (id,name,tran_dt,address)
```

```
    values ('100001','杨小华','01APR2011:10:02:23'dt,'北京王府井百货大楼');
    insert   into   jxsj.cust_inf   (id,name,tran_dt,address)
    values ('100002','高洪为','02APR2012:21:11:15'dt,'北京西城区商场');
    insert   into   jxsj.cust_inf   (id,name,tran_dt,address)
    values   ('100003','刘小青','03APR2012:11:33:28'dt,'山东济南图书城');
    insert   into   jxsj.cust_inf   (id,name,tran_dt,address)
    values   ('100004','马小虎','04APR2012:09:16:36'dt,'上海商城');
    insert   into   jxsj.cust_inf   (id,name,tran_dt,address)
    values   ('100005','张太华','05APR2012:07:06:56'dt,'天津百货公司');
    insert   into   jxsj.cust_inf   (id,name,tran_dt,address)
    values   ('100006','董小峰','06APR2012:22:08:21'dt,'山东济宁');
quit;
*取符合条件的交易数据;
%let   sj=%sysfunc(mdy(04,03,2012));   /*定义了宏变量，此值为日期值*/
%macro   custsj(v_dt);   /*定义宏过程 custsj，并定义宏参数 v_dt*/
%let   vdt=%sysfunc(putn(&sj,yymmddn8.));   /*对时间转换，生成 yyyymmdd 格式*/
data   cust_&vdt.;   /*数据集名为 cust_与传递的宏变量的组合生成按日期的数据*/
  set   jxsj.cust_inf (where=(datepart(tran_dt)>&sj));   /*取符合 where 条件的数据*/
    tran_date=datepart(tran_dt);
    format   tran_date   yymmdd10.;
run;
%mend   custsj;
%custsj(&sj);   /*调宏过程，并传达实参&sj*/
```

【程序解读】

1）proc sql noprint；语句是调 SAS 内部的 SQL 过程。此过程是 SAS 系统处理 SQL 语言用到的过程，其功能已经封装，用户只需要调用就可以。noprint 的选项告诉 SAS 系统不打印输出到输出窗口，提高了效率。

2）create table jxsj.cust_inf (id char(6) label='客户号', name varchar(10) label='客户姓名', tran_dt num format=is8601dt20. label='交易时间', address varchar(100) label='交易地点');这个语句为 SQL 语言中的创建表语句，重点学习此语句中的时间字段的定义以及标签的定义说明用法。“tran_dt num format=is8601dt20. label='交易时间'”定义的表变量时间为日期+时间的变量，定义为 num，format= is8601dt20.定义了此变量的输出显示格式。

3）insert into jxsj.cust_inf (id,name,tran_dt,address)

values ('100001','杨小华','01APR2011:10:02:23'dt,'北京王府井百货大楼');此语句为 SQL 语言中插入数据到表的语句，重点学习时间变量插入到目标表 cust_inf 时的 tran_dt 时的书写方式，'01APR2011:10:02:23'dt 日期时间值后面的 dt 表示为日期时间类型。

4）%let sj=%sysfunc(mdy(04,03,2012));此语句为定义宏变量语句，实际开发中为了应用灵活性，经常改动的变量一般用先定义宏变量的方式，其他需要引用此变量的地方只需要引用就可以，而不要修改每一个引用此变量的地方。对应宏变量如果需要调内部的宏函数需要先通过=%sysfunc()这个系统宏函数才能调用 SAS 系统内部的宏函数，本案例中引用宏函数 mdy()处理日期函数。

5）%macro custsj(v_dt);语句为定义宏过程语句，%macor 宏过程定义开始标志，custsj

为自己定义的宏过程名，v_dt 为用户定义的宏参数，在引用时由实参传递给形参。

6）%let vdt=%sysfunc(putn(&sj,yymmddn8.));此语句为对日期的特殊处理，函数 putn()把已经定义的宏变量 sj 转换为数值类型，输出格式为 yymmddn8.表示的日期格式，即 yyyymmdd 表示的日期形式。

7）data cust_&vdt.;此语句为宏过程引用的数据步开始语句，定义的数据集名为 cust_&vdt，可以看出这个数据集名字是变动的，根据引用的宏变量名&vdt 而变动。

8）set jxsj.cust_inf(where=(datepart(tran_dt)>&sj));此语句为数据步语句，通过 set 语句读取逻辑库 jxsj 下的表 cust_inf。注意，学习该语句后面的 where 条件语句，在条件过滤中引用函数 datepart(tran_dt)取出数据集中变量 tran_dt 的日期部分，把大于&sj 变量的日期的数据取出来。

9）tran_date=datepart(tran_dt);数据步中直接引入了一个新变量，把 datepart(tran_dt)函数处理的值赋值给 tran_date。

10）format tran_date yymmdd10.;foramt 语句定义变量 tran_date 的输出格式为 yymmdd10.。

【提示】 该案例重点学习 SQL 过程中日期时间的处理方式、宏变量的引用、宏过程中数据步过滤条件 where 的应用。

3.1.2 SET语句

SET 语句用在数据步，从一个或几个已经存在的数据集中读取数据，对生成的一个或多个数据集进行处理，可以对多个数据集进行复制或纵向合并。在读取数据集时严格按照数据集动态生成机制处理数据集，每一个观测先读入到 PDV 指针中，多个数据集需要多个 PDV 指针控制，处理机制不变。

语法：SET <数据集名 1> <选项>　<数据集名 2 > <选项>…;

功能：复制数据集或纵向合并数据集。

表 3-1　SET 语句功能说明

对　应　项	说　　明
SET	SET 关键字，实现复制数据集或纵向合并数据集
数据集名	指定复制或合并的数据集名
选项	如 keep=取保留变量、nobs=变量名，记录数据集的总观测数，赋值给此变量

【提示】 SET 语句处理数据集时是先对原数据集读取，把读取的数据逐条放到 PDV 中，然后输出到一个新数据集中，占用了两个存储空间（一个原数据集存放空间和一个新数据集存放空间），直到数据处理完成，才删除原始数据集。SET 语句处理数据集的过程中产生了一个副本数据集。

【例 3.7】 对例 3.1 数据集 students 处理，取 students 数据集中的一条记录，保留字段 name、sex 和 address，生成数据集 stu。

```
data  stu;
  set  students  (keep=name  sex  address  obs=1 ) ;
  /* set 语句读数据集 students,括号内 keep=选项语句取保留的字符，obs=1 取数据集的第一条记录*/
run;
```

【程序解读】

set students(keep=name sex address obs=1) ;此语句用在数据步中，通过 SET 语句读取已经生成好的数据集 students。此数据集由例 3.1 生成，数据集存储在 work 临时逻辑库中，set 语句中具有的选项语句需要用括号括起来。keep=取保留的字段，实际应用开发中做大数据量的数据处理时经常需要此语句，把需要的字段从数据集中取出来，从而提高了数据读取的效率。obs=1 取数据集的第一条观测记录。

【例 3.8】 数据集的纵向合并，对例 3.1 进行改造。一个学校有 3 个班级，3 个班级的数据集分别为 stu1、stu2、stu3，学校教务处需要一个汇总的数据集 sum_students。

```
/*每一位数据处理人员生成各自班级的数据*/
data    stu1 (keep=name sex class address);
/*数据集 stu1*/
   length     address $30.;
/*对应字符类型的变量其长度超过默认长度 8 个字符要通过 length 语句先定义变量*/
input   id   name   $   sex   $   class   address;/*定义变量，其中 address 为 length 语句中定义好的变量*/
cards;/*读 cards  后的数据*/
1001  高溪红  F 1  北京市西城区广外大街
; /*数据块结束这里要有个分号*/
run;
*数据集 stu2;
data    stu2 (keep=name sex class address);
   length     address    $30.;
   input    id    name    $    sex    $    class    address;
cards;/*读 cards  后的数据*/
1002  张明明  M 2  北京市东城区
;
run;
*数据集 stu3;
data    stu3 (keep=name    sex    class    address);
   length     address    $30.;
   input    id    name    $    sex    $    class    address;
cards;
1003  刘小红  M 2  北京市丰台区
;
run;
*汇总数据集 sum_students;
data    sum_students;
   set    stu1    stu2    stu3;/*读取 3 个数据集，合并生成数据集 sum_students*/
run;
```

【程序解读】

本案例分别由不同的人生成了 3 个班级的数据集，学校业务需求是汇总 3 个班级的数据集，生成 sum_students 数据集，属于数据集的纵向合并。语句“set stu1 stu2 stu3”为纵向合并数据集语句。

if 语句与 where 语句在 SET 语句中的应用与比较。

【提示】 对于条件过滤数据集的 SAS 程序，if 和 where 语句虽然实现的功能相同，但在效率方面，where 语句的效率最高，尤其是大数据量，更能看出 where 语句执行的时间远远小于 if 条件过滤执行的时间。

它们的区别如下：

if 语句过滤是后过滤，数据集中数据先读入到程序数据向量 PDV 中，放入开辟的 PDV 缓冲区，然后再判断过滤。

where 语句是把数据集中的数据先根据 where 条件进行过滤，把符合条件的数据取出来然后再存入 PDV 区，最终放入到数据 PDV 缓冲区中的数据是条件判断完的数据。

通过分析可以看出，where 语句的效率比 if 语句的效率高，因此实际开发和应用中能用 where 语句过滤的就不要用 if 语句过滤。

【例 3.9】 根据条件删除数据集。学习 where 条件语句与 if 条件语句的过滤方式和处理机制。

```
*数据集 stu;
data   stu   (keep=id name sex class );
  length     address $30.;
  input   id   name   $   sex   $   class   address;
cards;   /*读 cards  后的数据*/
1001  高溪红  F 1  北京市西城区广外大街
1002  张明明  M 2  北京市东城区
;
run;
*where 条件语句实现过滤取数据集 stu 中 id=1001 这条数据;
data   cz;
  set   stu;
  where   id=1001;   /*where 条件语句*/
run;
*if 条件语句实现过滤取数据集 stu 中 id=1001 这条数据;
data   cz2;
  set   stu;
  if     id=1001;   /*if 条件语句*/
run;
```

【程序解读】

1）cz 数据集是 where 条件语句生成的数据集。

2）cz2 数据集是 if 条件语句生成数据集。

【提示】 对比可以发现两个数据集 cz 和 cz2 数据一样，只是过滤条件语句不同。对于大数据量的数据处理才能明显看出 where 语句的优势。

【例 3.10】 二次处理数据集，取数据集的第一条记录。

```
*文件数据之间有分隔符文件读取;
%let   dir= D:\jx\dq;   /*定义外部文件路径*/
%let   gsm=.dat;
%let   filename= "&dir&gsm";
libname   jx 'd:\jx';   /*定义逻辑库*/
```

```
data   jx.dq;   /*数据集存储到指定逻辑库*/
       Infile   &filename   dlm='|'   dsd   missover ;
/*此处用 dlm='|'分隔符参数读取分隔文件*/
       input                  quhao              :$3.
/*格式修饰符":"定义变量，遇到分隔符此列读取结束*/
                              youbian          :6.
                              address          :$12.

                         ;
run;
*根据业务需求二次数据处理，取数据集第一条记录，其他输出到 other_dq;
data   jx.first_dq   jx.other_dq;
        set   jx.dq;
       if _n_=1   then   output   jx.first_dq ; /*取第一条记录输出到数据集 jx.first_dq */
            else    output   jx.other_dq;   /*不是第一条记录的输出到数据集 jx.other_dq */
 run;
```

【程序解读】

1）"if _n_=1 "条件语句利用了 SAS 内部的记录条数变量指针_n_，取_n_=1 为取第一条记录。条件成立执行"then output jx.first_dq"语句，通过 output 输出到数据集 jx.first_dq。

2）"else output jx.ohter_dq"表示当 if 语句不成立时执行此语句，输出到 jx.other_dq 数据集。

【提示】 对于条件语句 if，如果输出到其他数据集一定要加 output 语句。

【例 3.11】 do…end 语句在数据集二次处理中的应用。

```
*文件数据之间有分隔符文件读取;
%let   dir= D:\jx\dq;   /*定义外部文件路径*/
%let    gsm=.dat;
%let    filename= "&dir&gsm";
libname   jx   'd:\jx';   /*定义逻辑库*/
data   jx.dq;   /*数据集存储到指定逻辑库*/
       Infile   &filename   dlm='|'   dsd   missover ;
/*此处用 dlm='|'分隔符参数读取分隔文件*/
       input                  quhao              :$3.
/*格式修饰符":"定义变量，遇到分隔符此列读取结束*/
                                    youbian        :6.
                                    address            :$30.

                         ;
run;
*根据业务需求二次数据处理，取数据集第一条记录，对 address 变量进行处理;
data   jx.doend;
     set   jx.dq;
length   v_youbian   $10.;
if   _n_=1   then   do;/*取第一条记录*/
v_youbian='北京';
address='北京地区'||trim(address);
```

```
/*if 语句中 do…end 语句对字段处理*/

end;
   else    do;
          v_youbian='其他邮编';
      address='其他'||trim(address);
       end;
 run;
```

【程序解读】

1）if 语句里嵌套 do…end 语句，对于多个变量的处理需要放在 do…end 语句中才可以。

2）else 里语句中也嵌套了一个 do…end 语句，处理多个变量。

【提示】 对数据集中的多个变量，如果满足条件就进行特殊处理，需要 if 语句中嵌套 do…end 语句。do…end 语句是执行语句块语句，每一个 do 语句都对应一个 end 结束标志语句。

3.1.3 MERGE语句

MERGE 语句将两个或多个数据集进行横向合并。所谓横向合并是指两个不同的数据集拼接在一起。在实际应用中可能一个表的数据列变量太多，可以把数据存储在两个数据集中，通过 MERGE 合并，得到完整的数据集。

语法：MERGE <数据集名 1> <选项>　<数据集名 2 > <选项>…;

功能：实现数据集的横向合并。

表 3-2　MERGE 语句说明

对　应　项	说　　明
MERGE	MERGE 实现数据集横向合并
数据集名	指定合并数据集名字
选项	常用选项，如 keep=、drop=、rename=、where=、in=等

【提示】 MERGE 与 SET 语句的区别：MERGE 语句实现横向合并；　SET 语句是将两个或多个数据集纵向合并，SET 语句还有复制数据集的功能。

MERGE 语句在横向合并数据集时分为一对一合并和匹配合并两种。

1）MERGE 一对一横向合并数据集：所谓横向合并是指将两个或多个数据集拼接成一个数据集，将两个或多个数据集中的第一条观测记录合并成新数据集的第一条观测记录，第二条观测记录合并成新数据集的第二条观测记录，依次类推，没有的用缺失值替代。

2）MERGE 语句匹配合并：根据 BY 语句指定的公共变量的值实现横向合并。此处的 BY 语句中的变量相当于一个表的主键，也就是根据主键横向合并，要先对公共变量进行排序，然后再进行合并。

【例 3.12】 学生信息表 inf_stu 为学生信息数据集，学生成绩表 score_stu 为学生期末考试成绩，根据需求需要得到一个学生信息与成绩的数据集 inf_score，保留字段为 id(学号)、姓名和总成绩。

```
data    inf_stu;
```

```
  length    address $30.;
/*对应字符类型的变量的长度超过默认长度 8 个字符，要通过 length 语句先定义变量*/
  input   id   name   $   sex   $   class   address;
  /*定义变量，其中 address 为 length 语句中定义好的变量*/
cards;       /*读 cards  后的数据*/
1001  高溪红  F 1  北京市西城区广外大街
1002  张明明  M 2  北京市东城区
;
run;
data     score_stu;
  input   id   score;
cards;
1001 600
1002 890
;
run;
proc   sort   data=inf_stu;            /*对数据集排序*/
  by   id;
run;
proc   sort   data=score_stu;          /*对数据集排序*/
  by   id;
run;
*根据学号横向合并数据集 inf_stu 和 score_stu，生成数据集 inf_score;
data   inf_score   (keep=id name score);          /*keep=语句取保留字段*/
  merge   inf_stu   score_stu;
  by   id;                             /*根据学号横向合并*/
run;
```

【程序解读】

1）MERGE 语句横行合并时要先对两个数据集排序。此程序通过 SAS 的 sort 过程对数据集根据学号 id 按默认升序排序。

2）“merge   inf_stu   socre_stu”语句根据 id 合并数据集。

【例 3.13】 MERGE 语句查找两个数据集中相同的数据。

```
data     stu1;
  length    address $30.;
/*对应字符类型的变量的长度超过默认长度 8 个字符要通过 length 语句先定义变量*/
  input   id   name   $   sex   $   class   address; /*定义变量，其中 address 为 length 语句中定义好的变量*/
cards;   /*读 cards  后的数据*/
1001  高溪红  F 1  北京市西城区广外大街
1002  张明明  M 2  北京市东城区
;
run;
data     stu2;
  length    address $30.;
/*对应字符类型的变量的长度超过默认长度 8 个字符，要通过 length 语句先定义变量*/
```

```
    input  id  name  $  sex  $  class  address;  /*定义变量,其中 address 为 length 语句中定义好的变量*/
  cards;  /*读 cards 后的数据*/
  1001 高溪红 F 1 北京市西城区广外大街
  1003 杨小红 M 3 北京市丰台区
  ;
  RUN;
  data    public;
    merge  stu1(in=a)  stu2(in=b);  /*利用 SAS 系统的内部 in=变量，标识信息来自哪个数据集*/
      if a and b ;          /*条件 a and b 表示查找数据，即在 stu1 数据集也在 stu2 数据集的公共数据*/
      by   id;              /*根据 id 比对*/
  run;
```

【程序解读】

1）MERGE 语句中利用 SAS 系统的内部 in=变量，标识信息来自哪个数据集，根据条件 if a and b 查找出两个数据集的公共部分。

2）“by id”语句是必需的，告诉 MERGE 语句根据 id 匹配查找。

【例 3.14】 对例 3.13 进行改造，MERGE 语句查找两个数据集中不同的数据。

```
  data   other;
  merge  stu1(in=a)  stu2(in=b);   /*利用 SAS 系统的内部 in=变量，标识信息来自哪个数据集*/
  if  ^a and  b ;        /*条件 ^a and b 表示查找数据不在 stu1 数据集，在 stu2 数据集的数据*/
  by  id;                /*根据 id 比对*/
  run;
```

【程序解读】

“if ^a and b ;”语句中^a 表示条件为不在 stu1 数据集，b 表示在数据集 stu2 中的数据，查找出两个数据集中的不同数据。

3.2 数据集应用案例

灵活运用数据步才能综合处理各类数据。实际开发中根据业务需求需要对各类数据综合处理，而不是单独的，简单的数据处理。

3.2.1 数据集条件过滤

在实际开发中经常遇到各类数据文件，而不是 CARDS 语句或 DATALINES 语句块数据。因此，学习 SAS 对各类数据文件过滤的方法是数据处理的关键点。本节通过几个经典的案例进行详细的讲解和分析。

【例 3.15】 @指针应用，过滤数据控制。取出卡类型为 138 的数据，数据文件存储在 d:\jx\custer.dat。

```
   %let    dir= 'd:\jx\custer.dat';
   filename    sj   "(&dir)";
   libname   jx    'd:\jx';
  data     jx.card138;
```

```
        infile    sj   dsd   missover;
        input   @20    card_type   $3.
                @;  /*行控制指针符号，使数据取到 card_type 之后执行下面的 if 语句，做判断*/
         if   card_type='138'   then   do;    /*取 card_type='138'类型的条件数据*/
             input    @1   qh       $3.
                      @4   card_nm   $16.
                      @23 name $8.
                      @31 address :$20.
                      ;
                        output   jx.card138;   /*output 语句把取出的数据输出到此数据集*/
                    end;
run;
```

【程序解读】

重点理解过滤条件语句中行控制指针@的应用。“input @20 card_type $3. @;”这条语句先取外部数据文件 custer.dat 中的 card_tyep 列对应的数据，取到之后行控制指针@使其指针停留在当前行，执行后面的“if card_type='138' then do;”条件判断语句，把符合此条件的这条记录对应的数据的其他列信息取出来，然后返回上面的“input @20 card_type $3. @;”语句继续执行取其他行数据，并做判断。

【提示】 这个程序经常用到。实际的开发中遇到一个文件有上百万条数据是常见的，为提高程序的效率，先对文件进行过滤，只取符合条件的数据，而不是一开始就把外部数据文件转换成 SAS 数据集再进行过滤，这是不可取的一种方式。虽然也能实现相同的功能，但效率低，需要很长的时间。

【例 3.16】 删除数据集中不需要的数据，对例 3.15 进行改造，删除卡类为 138 的数据，建立数据集 other138。

```
%let    dir= 'd:\jx\custer.dat';
filename    sj    "(&dir)";
libname   jx     'd:\jx';
data   jx.other138;
        infile    sj dsd missover;
        input   @20    card_type   $3.
                @; /*行控制指针符号，使数据取到 card_type 之后执行下面的 if 语句，做判断*/
         if   card_type='138'    then   delete;/* delete 删除语句删除 138 对应的数据行*/
         if   card_type^='138'   then   do;    /*取 card_type='138'类型的条件数据*/
             input    @1   qh       $3.
                      @4   card_nm   $16.
                      @23 name $8.
                      @31 address :$20.
                ;
                        output    jx.other138;
                    end;
run;
```

【程序解读】

此程序通过“if card_type='138' then delete;”语句先执行删除符合条件的数据，然后执

行“if　card_type^='138'　then do;”语句，把不是 138 的数据取出来，执行 do…end 语句输出其他列数据。

【提示】 if 语句中可以执行删除语句，直接删除就可以。

【例 3.17】 对于有文件头信息的，从文件中的第二条数据开始读取，通过宏过程实现。

```
%let    dir= 'd:\jx\inf_cust.dat';/*文件存储路径宏变量*/
filename    sj   "(&dir)" ;/*定义逻辑文件名*/
libname   csj   'd:\jx';/*定义存储数据集逻辑库*/
%macro   loadfile(v_lib);   /*宏过程开始，并定义宏变量*/
 data   &v_lib.inf_custer;
        Infile   sj   firstobs=2   end=final   length=length;   /*firstobs=2 表示从文件的第二条数据*/
/*开始读取*/
         input   @1 qh 3.
                 @4 card_nm $16.
                 @20 card_type $3.
                 @23 name $8.
                 @31 address :$20.
                   ;
  run;

%mend;
%loadfile(csj.); /*调用宏过程，传递实参逻辑库名，注意这里实参传逻辑库名加个.*/
```

【程序解读】

宏过程 loadfile 中“infile sj firstobs=2 end=final length=length”语句中的 firstobs=2 表示从外部文件的第二条记录开始读取。对于有文件头信息的需要用此程序的方法读取外部文件。

【例 3.18】 模糊查询条件的宏应用。通过 SAS 系统内部的宏变量参数，实现查找北京、上海和属于山东省的城市对应的数据，生成数据集 city。

```
data   city_inf;
length   city $30.;
input   id   city   $;
cards;
1001  上海市南京路
1002  北京西城区
1003  上海市浦东区
1004  天津滨海新区
1005  北京东城区
1006  山东济南
1007  山东济宁
1008  北京市丰台区
;
run;
/*通过宏 indexs 实现模糊查找*/
%macro   indexs /parmbuff;   /*通过宏系统的内部参数宏变量 parmbuff 接收传递的变量信息*/
   %local   num   dsname;   /*定义局部变量*/
```

```
    %let   num=2;   /*变量赋值*/
    %let   dsname=%scan(&syspbuff,&num);
    /*把 parmbuff 的信息传递给宏系统内部宏变量 syspbuff，scan 函数实现查找功能，*/
    /*这里 num=2，从 syspbuff 变量的第二个变量开始查找*/
    %do   %while(%quote(&dsname)   ne %quote());   /*判断循环语句，如果变量 dsname 不为空*/
          %if   &num>2   %then   %do;
                   or
           %end;
           index(%scan(&syspbuff,1),"&dsname")>0 /*从第一个变量查找，搜索 dsanme 变量的位置*/
       %let   num=%eval(&num+1);   /*num 变量加 1*/
       %let   dsname=%scan(&syspbuff,&num);   /*从第 num 位查找，并负责给 dsname*/
    %end;
%mend ;

data   city;
set   city_inf;
where   %indexs(city,北京,上海,山东);
/*调用宏 indexs，模糊查询 city 变量为北京、上海和属于山东省的城市的数据*/
run;
proc   print   data=city;
run;
```

输出窗口显示信息如图 3-1 所示。

```
输出 - （无标题）
                 SAS 系统          2012年04月27日 星期五

       Obs    city              id

         1    上海市南京路      1001
         2    北京西城区        1002
         3    上海市浦东区      1003
         4    北京东城区        1005
         5    山东济南          1006
         6    山东济宁          1007
         7    北京市丰台区      1008
```

图 3-1　查找信息显示

日志显示 where 条件后的信息为生成了 3 个条件的或语句，如图 3-2 所示。

```
日志 - （无标题）
58           %let dsname=%scan(&syspbuff,&num);/*从第num位查找，并负责给dsname*/
59
60        %end;
61     %mend ;

62     data city;
63     set city_inf;
64     where %indexs(city,北京,上海,山东);
65     run;

NOTE: 从数据集 WORK.CITY_INF. 读取了 7 个观测
      WHERE (INDEX(city, '北京')>0) or (INDEX(city, '上海')>0) or (INDEX(city, '山东')>0);
NOTE: 数据集 WORK.CITY 有 7 个观测和 2 个变量。
NOTE: “DATA 语句”所用时间（总处理时间）：
      实际时间            0.01 秒
      CPU 时间            0.00 秒
```

图 3-2　查找宏过程日志显示信息

【程序解读】

本程序通过编写 indexs 宏过程和 SAS 系统内部宏变量的应用以及宏函数结合的应用，实现模糊查找。

对上面的模糊查找也可以写 SQL 语句实现，只是这种方式不简便，需要写很多语句。对例 3.18 通过 SQL 语句实现同样的功能。

【例 3.19】 通过 SQL 语句实现模糊条件查找，对例 3.18 进行改造。

```
PROC   sql; /*调 SQL 过程*/
select   * from    city_inf   where   city   like '%北京%' or city like '%上海%' or city like '%山东%';
/*where 条件中 city like 语句查找出 city 办理包含北京或上海或属于山东的城市*/
quit;
```

【程序解读】

where 条件中 like 语句实现模糊查找，其中%（百分号）表示任意字符。例如，“%北京%”表示不管“北京”前面和后面是什么字符，只要包含“北京”就符合条件。

【例 3.20】 根据条件，实现多个数据集用同一个宏过程进行打印的功能。

```
data   city_inf1;
length   city   $30.;
input   id   city   $;
cards;
1001  上海市南京路
1002  北京西城区
;
data   city_inf2;
length   city   $30.;
input   id   city   $;
cards;
1003  上海市浦东区
1004  天津滨海新区
;
data   city_inf3;
Length   city   $30.;
input id city   $;
cards;
1005  北京东城区
1006  山东济南
1007  山东济宁
1008  北京市丰台区
;
data   custer;
input   id   name   $;
cards;
1001  刘小红
;
run;
```

```
/*实现数据集的打印功能，可以传递多个数据集*/
%macro   printz/parmbuff;
    %put   Syspbuff contains: &syspbuff;   /*系统内部宏变量 syspbuff 记录了 dsanme 变量的信息*/
    %let   num=1;
    %let   dsname=%scan(&syspbuff,&num);
    %do   %while(&dsname ne);
        proc   print   data=&dsname;   /*打印数据集名 dsname 宏变量对应的数据集*/
        run;
        %let   num=%eval(&num+1);   /*变量加 1 取下一个数据集的名字*/
        %let   dsname=%scan(&syspbuff,&num);
/*从第 num 变量开始查找并把查找到的内容赋值给变量 dsanme*/
    %end;
%mend   printz;
/*调用宏过程 printz，传递数据集名实参 city_inf1,city_inf2,city_inf3,custer 给 SAS 内部宏变量 parmbuff*/
%printz(city_inf1,city_inf2,city_inf3,custer);
```

【程序解读】

这个程序主要学习 SAS 系统内部宏变量 parmbuff 和 syspbuff 两个内部宏变量的联合应用的功能，实现根据循环条件取数据集名，同时应用 SAS 系统内部函数，实现了强大功能的封装。

3.2.2　CALL子程序数据步应用

【例 3.21】 CALL 子程序数据步应用。

```
%macro   callpro(v_param);
    data   _null_;
        x = 'we are friends';
        z=&v_param;
        call   symput('v_var',x);/*把 x 变量数据赋值给变量 v_var*/
%mend   callpro;
 %callpro(1)
run;
/*  上面 CALL 子程序语句中的变量 v_var 在下面引用*/
data temp;
    y="&v_var";   /*引用上面的变量 v_var*/
run;
proc   print   data=temp;
run;
```

【程序解读】

1）通过 CALL 子程序调 symput 函数，把 X 变量值赋值给 v_var。

2）下面的 data temp; y="&v_var";数据步引用宏变量值给数据步 y 变量。

【例 3.22】 条件语句中执行 CALL 子程序案例。

```
data   cust_inf;
```

```
    input   dept   $ name   $   salary @@;
    datalines;
  BI  高明  18000 gov  刘小红  16000
  pulic  董小云  9000 private  马西名  8000
;
proc   means   data=cust_inf   noprint;
    class   dept;
    var   salary;
    output   out=analy   sum=s_salary;
run;

proc   print   data=analy;
    var   dept   s_salary;
    title   "统计信息";
    title2   "对部门统计分析";
run;
/*if  条件语句调 call 子程序的应用*/
data _null_;
    set   analy;
    if   _n_=1   then   call   symput('s_total',s_salary); /*条件满足执行 call 子程序语句*/
    else   call   symput('v_str',dept);
run;
data   ana_temp;
v_total=&s_total;     /*引用 call 语句中的变量 s_total*/
run;
```

【程序解读】

1）理解数据步 if 条件语句调 call 子程序的运用。

2）“if _n_=1 then call symput('s_total',s_salary);else call symput('v_str',dept);”语句在数据步处理变量的赋值传递，条件满足传递给一个新变量。

3.2.3 数据集输出应用

数据集输出是对数据步处理数据后生成的数据集。数据集输出按输出逻辑库划分分为输出到临时逻辑库和输出到永久逻辑库。如果数据集处理的结果只是供临时应用，可以将数据集输出到临时逻辑库，从而节省了存储空间；如果数据集需要永久保留就要将数据处理结果输出到永久逻辑库。

1．输出数据集到临时逻辑库

语法格式：[逻辑库名].数据库名。

【说明】 SAS 系统数据集输出到临时逻辑库默认是 work 逻辑库，逻辑库名可以省略。直接写数据集名就可以。

【例 3.23】 数据集输出到临时逻辑库 work 库。

```
*数据集输出到临时逻辑库，逻辑库名 work 可以省略，默认是到临时逻辑库;
data   work.custers;/*数据集 custers 输出到临时逻辑库 work*/
```

```
    input   dept   $   name   $   salary @@;
    datalines;
  BI  高明  18000 gov  刘小红  16000
  pulic  董小云  9000 private  马西名  8000
;
run;
```

【程序解读】

数据步输出数据集到临时逻辑库 work 下，其逻辑库名可以省略。此程序数据集 custgers 生成到 work 逻辑库下。

【例 3.24】 一个数据步根据条件生成多个数据集到临时逻辑库 work 库。

```
  *定义 3 个数据集，输出到临时逻辑库;
data   custbi   custpulic   custother ;   /*数据集 custbi custpulic custother*/
    Input   dept   $   name   $   salary @@;
    if   dept   eq   'BI'   then   output   custbi;
    /*if 语句条件判断，根据部门 dept 判断，output 语句后面为输出到数据集名*/
      else   if   dept   eq   'pulic'   then   output   custpulic;
/*pulic 部门的输出到数据集 custpulic*/
      else   output   custother;   /*把其他部门输出到数据集 custother*/
    datalines;
  BI  高明  18000 gov  刘小红  16000
  pulic  董小云  9000 private  马西名  8000
;
run;
```

【程序解读】

1）数据步语句“if   dept eq 'BI' then output custbi;”表示如果 dept 变量的值为“BI”,输出数据集到 custbi。

2）数据步语句“else if dept eq 'pulic' then output custpulic;”表示如果 dept 变量的值为“pulic”，输出数据集到 custpulic。

3）数据步语句“else   output custother;” 表示如果上面两个条件都不成立，输出数据集到 custother。

【提示】 对 datalines 语句块进行数据处理时，如果需要条件语句必须把条件语句写在 input 语句与 datalines 语句之间。

对于 infile 语句读入外部数据文件，其条件语句放置的位置与例 3.24 是不同的。

【例 3.25】 infile 语句读入外部文件，根据条件生成多个数据集到临时逻辑库。

```
   %let    dir= 'd:\jx\custer.dat';
filename    sj    "(&dir)";
data   cust138   othercust;   /*定义两个数据集 cust138 和 othercust*/
            infile    sj   dsd   missover;    /*读外部数据文件*/
            input    @1   qh       $3.
                     @4   card_nm   $16.
                     @20 card_type $3.
```

```
                    @23 name $8.
                    @31 address :$20.
                    ;
    if    card_type='138'    then    output    cust138;
    /*if 条件语句 card_type='138'数据输出到 cust138 数据集*/
        else    output    othercust;
    /*其他 card_type 不是'138'数据输出到 othercust 数据集*/
run;
```

【程序解读】

1）“if　card_type='138'　then output cust138;”语句表示 card_type 为'138'的输出到数据集 cust138。

2）“else output othercust;”语句表示 card_type 不为'138'的输出到数据集 othercust。

【提示】 对比例 3.24 和例 3.25，读入语句块和读入外部文件 if 语句放的位置的不同。

2．输出数据集到永久逻辑库

对应数据集输出到永久逻辑库需要先创建永久逻辑库，然后通过数据步引用永久逻辑库名把相应的数据集存储到指定的物理路径下。

引用永久逻辑库方式：永久逻辑库名.数据集名。

对应永久逻辑库首先要定义永久逻辑库：

Libname 逻辑库名 物理路径;

【例 3.26】 将例 3.25 生成的数据集存储到物理路径为 d:\jx 的文件夹中。

```
%let     dir= 'd:\jx\custer.dat';
filename     sj    "(&dir)";
libname    jx     'd:\jx';    /*定义永久逻辑库 jx*/
data    jx.cust138    jx.othercust;     /*数据步引用永久逻辑库 jx*/
            infile     sj    dsd    missover;
            input    @1       qh             $3.
                     @4       card_nm        $16.
                     @20      card_type      $3.
                          @23     name           $8.
                          @31     address      :$20.
                     ;
    if    card_type='138'    then    output    jx.cust138;
    /*if 条件语句 card_type='138'数据输出到 jx.cust138 数据集*/
        else    output    jx.othercust;
    /*其他 card_type 不是'138'数据输出到 jx.othercust 数据集*/
run;
```

【程序解读】

1）“if　card_type='138'　then output jx.cust138;”语句表示符合条件 card_type 为 138 的输出到永久逻辑库 jx 所对应的物理路径下，数据集名为 cust138。

2）“else output jx.othercust;”语句表示 card_type 不为'138'的输出到永久逻辑库 jx 所对应的物理路径下，数据集名为 othercust。

3.2.4 数据集加密码应用

实际开发中对应重要的数据需要生成加密数据集，防止数据在传输过程中被截获而泄露数据信息。数据集加密是从数据安全的机制考虑。使用数据集的一方需要知道解密密码才可以应用数据集。

【例 3.27】 对生成的 a 数据集加密，写密码为“ycr”，另一用户更新时会弹出输入密码窗口。

```
/*对数据集加密码，其他用户如果不知道密码就只能读不能更新写操作表*/
data  a(write=ycr);/*对创建的数据集 a 通过 write=语句加入密码为“ycr”*/
input  x;
cards;
1
2
3
;
run;
proc  sql;/*调用 SQL 过程*/
  update  a  set x=0  where x=2;
quit;  /*SQL 过程对应结束标志 quit*/
```

【程序解读】

1）数据步语句“DATA a (write=ycr)”表示对数据集 a 通过 write=语句创建加密写密码 ycr，其他用户如需要更改此数据集 a，必须输入密码。

2）SQL 过程步中 update 语句更新 a 数据集，执行后会弹出如图 3-3 所示的交互对话框，需要用户输入密码才能更新 a 数据集。



图 3-3　输入密码对话框

【例 3.28】 对例 3.27 进行改造，更新数据集时防止出现交互对话框，直接把密码写在更新语句中。

```
  proc  sql;/*调用 SQL 过程*/
  update  a (write=ycr)  set x=0  where x=2;
/*update 语句中 a 数据集加入选项 write=ycr，告诉 SAS 系统更新密码权限*/
quit;  /*SQL 过程对应结束标志 quit*/
```

【程序解读】

语句“updata a (write=ycr)”直接写入密码选项参数，这样避免了交互对话框的弹出，直

接告诉 SAS 系统更新权限密码。

3.3 外部数据处理案例

应用中实际的数据处理大多数据都是外部数据文件，如各类文本文件、主机数据文件、各类数据库数据等外部数据，而不是 cards 语句或 datalines 语句后面的语句块形式出现的数据形式。因此学习本节外部数据文件处理的方法和技巧是实际开发和应用的需要。对于一个大型项目的开发，数据量都是在百万级以上，对外部数据处理的优劣直接影响到整体系统的效率。对外部数据处理的过程称为 ETL 过程（数据的过滤、转换和装载）。ETL 过程是大型数据仓库开发的核心。

3.3.1 TXT文件数据处理

TXT 格式文件是经常见到的文本文件。以.txt 为扩展名存储的文件，称为 TXT 格式文件。

【例 3.29】 读取 d:\jx\xs.txt 文件，创建数据集 xs。

```
*文件数据按列模式读取;
%let    dir= D:\jx\xs;   /*定义外部文件路径*/
%let    gsm=.txt;
%let    filename = "&dir&gsm";
libname   jx 'd:\jx';   /*定义逻辑库*/
data   jx.xs;   /*数据集存储到指定逻辑库*/
      infile   &filename   dsd   missover   ; /*infile 语句读入外部文件*/
      input      bh         1-10     /*列模式读取数据，从第一列到第 10 列为 bh 变量对应的数据*/
                 xm         $ 11-18
                 mobile    19-29
                 dx         $30-41
                ;
run;
```

【程序解读】

1）infile 语句读入逻辑名 filename 对应物理位置处的数据文件 d:\jx\xs.txt。

2）input 语句定义变量，数据文件中每一列对应一个变量，相应列的数据存储到对应变量中。

3）此处通过列模式读入变量对应的数据，列开始位置到列结束位置。

【提示】 对应上面的程序也可以通过绝对指针方式和列模式两种方式混合应用读入外部数据文件。

【例 3.30】 对例 3.29 进行改造，通过绝对指针和列混合模式两种方式同时运用读入外部数据文件。

```
*文件数据按列模式读取;
%let    dir= D:\jx\xs;   /*定义外部文件路径*/
%let    gsm=.txt;
%let    filename = "&dir&gsm";
```

```
libname  jx  'd:\jx';  /*定义逻辑库*/
data  jx.xsjd;  /*数据集存储到指定逻辑库*/
      infile  &filename  dsd  missover  ;
      input   @1  bh        10.  /*绝对指针@模式读取数据*/
                  xm            $ 11-18  /*列模式读取数据*/
                 @19  mobile   11.
                 @30  dx        $12.
              ;
run;
```

【程序解读】

1）input 语句中定义变量，通过@指针控制列开始位置，“@列开始位置 变量名 变量类型”。

2）此程序用了两种方式读外部数据，语句“xm $11-18”列模式读入外部数据文件对应列的数据。

3.3.2 Excel数据处理

Excel 文件也是实际开发中经常遇到的一类文件。对于 Excel 生成这类外部数据文件，一般的思路是先通过 SAS 过程步中的 IMPORT 过程生成 SAS 可以识别的数据集，然后再根据需求对生成的数据集进行处理，生成 SAS 可以识别的数据，运用数据步和 SAS 函数，根据需求对字段进行处理。

【例 3.31】 导入 Excel 数据 d:\jx\house.xls，生成 SAS 数据集，并取数据集的 9 条记录。

```
*通过 IMPORT 过程读取外部 Excel 文件建立数据集;
libname   jx  'd:\jx';
proc  import  out=jx.house  /*输出的数据集名*/
  datafile="d:\jx\house.xls";
/*要导入的 Excel 文件的完整路径与文件名，写清楚文件的扩展名*/
 sheet="fz";  /*指出电子表格中的那一个表单，就是表单名字*/
  getnames=YES;  /*指出第一行是否有字段名*/
run;
/*数据集二次处理*/
data  jx.house9;
  set  jx.house (obs=9);  /*obs=9 选项参数指定取数据集前 9 条记录*/
run;
```

【程序解读】

1）import 过程导入外部数据，out=选项指定输出数据集。

2）datafile=指定外部数据文件路径。

3）sheet=选项指定导入电子表格中的那个表单。

4）getnames=选项指定第一行是否有列名，有为 YES，否则为 NO。

数据步对数据二次处理，obs=9 取数据集前 9 条记录。

【提示】 datafile 和 table 不能一起用，如果指定是哪张表单可以用 range=或 sheet=选项语句，不要用 table=。

实际开发中经常遇到放在同一个文件夹下的同一类文件，对这类文件数据转换生成 SAS 数据集可以通过宏过程批量读入。

【例 3.32】 读入 d:\jx\ty 文件夹下的所有 Excel 文件，生成 SAS 数据集。

```
            *取路径下的所有文件名;
%let   dir=d:\jx\ty;
%macro   fetchfile(v_wz);
filename   wjlj    "%bquote(&v_wz.)";
data   filename;
   v_open=dopen('wjlj');
   if   v_open>0   then   do;
   v_num=dnum(v_open);
   do   i=1   to   v_num;
   v_tmp=cats("v_tmp",i);
   v_tmp=tranwrd(v_tmp,"","");
   v_mem=dread(v_open,i);
   output   filename;
   end;
end;
keep   v_mem   v_tmp;
 v_close=dclose(v_open);
run;
filename   wjlj   clear;
 %mend   fetchfile;
 %fetchfile(&dir);
%macro   readexcl(v_fname=,v_temp=);
Proc   import   out=&v_temp
   datafile="&dir\&v_fname"
   dbms=EXCEL replace;
   sheet="sheet1$";
   getnames=yes;
   mixed=no;
   scantext=yes;
   usedate=yes;
   scantime=yes;
run;
%mend   readexcl;
   data   _null_;
     set   filename;
call   execute('%readexcl(v_fname='||v_mem||',v_temp='||v_tmp||')');
run;
proc   datasets   library=work nolist;
delete   target;
run;
%macro   hbdata(v_tmp=);
proc   append   base=target   data=&v_tmp force;
```

```
run;
%mend   hbdata;
data   _null_;
  set   filename;
  call   execute('%hbdata(v_tmp='||v_tmp||')');
run;
%macro   deletmp(v_tmp=);
proc   datasets   library=work nolist;
delete   &v_tmp;
run;
%mend   deletmp;
data   _null_;
  set   filename;
call   execute('%deletmp(v_tmp='||v_tmp||')');
run;
```

【程序解读】

1）v_open=dopen（'wjlj'）；语句判断文件是否打开。

2）v_close=dclose（v_open）；语句关闭文件判断。

3）Proc　import　out=&v_temp 语句导入外部文件。

上面的程序是宏过程与数据步综合处理数据的应用。

3.3.3　CSV格式数据处理

CSV 格式的文件也是实际开发中经常遇到的一类文件。这类文件是带逗号分割的文本文件，文本文件存储的扩展名为.csv。

【例 3.33】 读入 d:\jx\tl 文件夹下的单个 lifecsv.csv 文件，生成 SAS 数据集。

```
  libname     jx 'd:\jx\tl';
proc   import   out= jx.flifecs
              datafile = "d:\jx\tl\lifecsv.csv"
              dbms=csv   replace;
       getnames =YES;
       datarow=3;   /*从文件的第三行读入*/
run;
```

【程序解读】

1）datafile=语句指定读入外部数据的物理路径。

2）dbms=csv 选项指定 SAS 系统能识别的 PC 文件标志。例如，对于有分割的文件，可以用.csv、dlm、jmp 。如果是读入表用 tab。

3）replace 选项表示对已经存储的数据集覆盖。如果不指定此选项，存储的数据集不覆盖。

4）datarow=指定从数据文件的第几行读入。

【提示】 CSV 和 txt 有相似的地方，需要 datarow=选项指定从第几行读入数据，不设置时默认从第二行开始读入，datarow=2。getnames=选项指定文件的第一行是否是列名，SAS

默认了 yes。

【例 3.34】 读入 d:\jx\tl 文件夹下的单个 lifecsv.csv 文件，生成 SAS 数据集。

```
libname     jx 'd:\jx\tl';
%macro   fetchonlyfile(v_filname) ; /*定义一个宏 fetchonlyfile*/
/*调用 IMPORT 过程导入外部数据文件，以传递变量的形式灵活读入*/
proc   import   out= jx.f&v_filname
                datafile= "d:\jx\tl\&v_filname..csv"
                dbms=csv   replace;
       getnames=yes;
       datarow=3;
run;
%mend   fetchonlyfile;
%fetchonlyfile(lifecsv);/*调用宏，并传递实参 lifecsv 给形参*/
```

【程序解读】

1）宏过程实现读入外部数据文件，把功能给予封装。通过传递参数给宏 v_filname。

2）%fetchonlyfile（lifecsv）语句调用宏过程，并传递实参 lifecsv 给形参 v_filname 变量。传递的是文件名。

3.3.4　DAT格式数据处理

DAT 格式文件和 TXT 文本文件的性质一样，只是 DAT 格式文件在实际开发中更经常遇到。这类文件存储的好处是其二进制文件数据。TXT 格式是字符文件。

【例 3.35】 对例 3.29 进行改造，读取 d:\jx\xs.dat 文件，文件扩展名为 dat 格式创建数据集 xsdat。

```
*文件数据按列模式读取;
%let    dir= D:\jx\xs;   /*定义外部文件路径*/
%let    gsm=.dat;
%let    filename = "&dir&gsm";
libname   jx 'd:\jx';   /*定义逻辑库*/
data   jx.xsdat;   /*数据集存储到指定逻辑库*/
       infile   &filename   dsd   missover   ; /*infile 语句读入外部文件*/
       input      bh        1-10     /*列模式读取数据，从第一列到第 10 列为 bh 变量对应的数据*/
                xm         $ 11-18
               mobile      19-29
               dx           $30-41
                 ;
run;
```

【程序解读】

1）infile 语句读入逻辑名 filename 对应物理位置处的数据文件 d:\jx\xs.dat。

2）input 语句定义变量，数据文件中每一列对应一个变量，相应列的数据存储到对应变量中。

3）此处通过列模式读入变量对应的数据，列开始位置到列结束位置。

【提示】 对应上面的程序也可以通过绝对指针方式和列模式两种方式混合应用读入外部数据文件。

【例 3.36】 对列 3.35 进行改造，通过绝对指针和列混合模式两种方式同时运用读入外部数据文件。

```
*文件数据按列模式读取;
%let    dir= D:\jx\xs;    /*定义外部文件路径*/
%let    gsm=.dat;
%let    filename = "&dir&gsm";
libname   jx 'd:\jx';   /*定义逻辑库*/
data   jx.xsjddat;   /*数据集存储到指定逻辑库*/
       infile   &filename   dsd   missover   ;
       input     @1   bh          10.   /*绝对指针@模式读取数据*/
                 xm                 $ 11-18   /*列模式读取数据*/
                @19   mobile     11.
                @30   dx          $12.
                 ;
run;
```

【程序解读】

1）input 语句中定义变量，通过@指针控制列开始位置，“@列开始位置 变量名 变量类型”。

2）此程序用了两种方式读外部数据，语句“xm $11-18”列模式读入外部数据文件对应列的数据。

【提示】 通过对比可以看出，DAT 格式与 TXT 格式的文件读取方式一样，只是文件扩展名格式不同。

3.3.5 关系数据库数据处理

实际业务开发中 SAS 系统经常与 Oracle 关系数据库交互应用，这也体现了 SAS 与其他数据库的交互能力。SAS 系统内部通过 SAS/Access 模块连接关系数据库。

【例 3.37】 通过 SAS 程序处理 rd_fgwj.dat 外部文件，并装载到 Oracle 数据库目标表 rd_test 中。

```
libname   sjk   oracle   user=chiran   password=chiran   path=orcl;
%let   f1 =d:\jx\rd_fgwj.dat;
%let   filrd = "&f1";
data   rdtest ;
Infile   &filrd   dlm='|'   lrecl=389   dsd   missover   firstobs=1   obs=3 ;
input   id                    :$8.
        type                  :$3.
        lxh                   :$32.
        lx_nane               :$40.
        c_type                :$3.
        ex_ins                :$30.
        sf                    :$1.
```

```
        sfs                 :$1.
        vf                  :$1.
        blk                 :$1.
        sfmo                :$1.
        sfzh                :$1.
        dte_rq              :yymmdd10.
        crlimit             :20.2
        using               :20.10
        pdc                 :20.10
        pdd                 :20.10
        score_c             :20.2
        score_d             :20.2
        score_scale         :$6.
        r_increase          :$4.
        l_increase          :20.2
        s_c_per             :20.2
        s_c_tem             :20.2
        score_re_code       :$3.
        score_dt            :$8.
        h_re_f              :$1.
        h_ref_c             :$6.
        debrecord           :$20.
;

run;
proc   append   base=sjk.rd_test
      ( bulkload=no
        dbsastype=(
     dte_rq              ='DATE'
     crlimit             ='NUMERIC'
     using               ='NUMERIC'
     pdc                 ='NUMERIC'
     pdd                 ='NUMERIC'
     score_c             ='NUMERIC'
     score_d             ='NUMERIC'
     l_increase          ='NUMERIC'
     s_c_per             ='NUMERIC'
     s_c_tem             ='NUMERIC'
      )
      NULLCHAR=NO /*告诉 SAS 系统缺失值是以 NULLCHARVAL=指定值替换*/
      NULLCHARVAL=" "
      )
      data=rdtest;   /*要装载的数据集*/
run;
```

【程序解读】

1）数据步首先对外部数据文件 rd_fgwj.dat 处理，生成 SAS 能够识别的数据集 rdtest。

2）proc append：过程步通过 append 过程把数据集 rdtest 装载到 Oracle 数据库表 rd_test。

3）bulkload=no：选项设置告诉 SAS 系统不用 Oracle 的 sql louder 方式装载，不指定此选项默认 Oracle 的 sql louder 方式装载。

4）dbsastype=：SAS 数据集中变量为数值类型和日期类型的必须通过此选项指定变量类型才能正确地装载到 Oracle 数据库对应的表中。

5）NULLCHAR=NO：告诉 SAS 系统缺失值是以 NULLCHARVAL=指定值替换。

6）NULLCHARVAL=" "：指定以空替换。

3.3.6　批量数据文件处理

对于实际应用开发而言数据量比较大，达到百万级记录以上的数据通常存储在不同的文件中，只是文件类型一样，对同一类型的数据文件可以采用批量导入数据文件的处理方式，从而提高了数据处理效率和简便性。

【例 3.38】 批量导入 d:\jx 文件夹下的所有 txt 格式文件，数据集存储到临时逻辑库。

```
    X   "dir d:\jx\*.txt /b >d:\jx\jx_qb";
/*x 为执行 Windos 命令，将 d:\jx 中的所有 txt 格式的文件的文件名输出到 jx_qb 文件中*/
%macro fetchfile(v_filname) ; /*定义一个宏 fetchfile*/
/*调用 IMPORT 过程导入外部数据文件*/
proc   import   out= work.f&v_filname
                datafile= "d:\jx\&v_filname..txt"
                dbms=tab   replace;
      getnames=yes;
      datarow=3;
run;
%mend fetchfile;
/*  通过数据步，执行 call execute 语句将 jx_qb 文件中的文件名依次作为宏 fetchfile 的参数*/
data     _null_;
         infile   "d:\jx\jx_qb";/*读外部文件 jx_qb*/
         input   v_str   :$60.;
         call   execute (compress('%fetchfile('||scan(v_str,1,'.')||');'));
/*调用宏过程 fetchfile，批量导入 txt 类型的所有外部数据，生产相应的数据集*/
run;
```

【程序解读】

1）通过 x 命令，执行“dir d:\jx*.txt /b >d:\jx\jx_qb”语句，通过 dir 命令切换到 d:\jx*.txt，通过管道符>把同类型的 txt 格式文件的文件名输出到 d:\jx 文件夹下的 jx_qb 文件中。

2）创建宏过程 fetchfile，调用 import 过程，执行文件导入命令。

3）数据步通过 call execute 语句调用宏过程 fetchfile，传递实参 v_str 的值给宏变量 v_filname。

3.3.7　宏过程数据处理

宏过程是把功能进行封装，就像 Oracle 数据库的存储过程一样。

【例 3.39】 宏过程实现外部数据处理，对例 3.36 进行改造。

```
*宏过程实现数据文件处理;
%macro   readfile(v_dir,v_type,v_name);
/*定义文件路径宏变量与文件名 v_dir，定义文件格式宏变量 v_type，定义数据集名宏变量 v_name*/
%let     dir=&v_dir.;   /*v_dir 引用*/
%let     gsm=.&v_type.;/*文件格式宏变量引用*/
%let     filename = "&dir&gsm";
libname    jx 'd:\jx';   /*定义逻辑库*/
data    jx.sj&v_name;    /*数据集存储到指定逻辑库，数据集名为 v_name 变量传递值*/
        infile   &filename   dsd   missover   ;
        input     @1 bh            10.   /*绝对指针@模式读取数据*/
                  xm               $ 11-18   /*列模式读取数据*/
                 @19    mobile     11.
                 @30    dx          $12.
                  ;
run;
%mend readfile;
%readfile( D:\jx\xs,dat,msjxsdat);/*调用宏过程，并传递实参*/
```

【程序解读】

1）%macro 是宏定义语句的开始，readfile 为宏名，v_dir、v_type 和 v_name 为宏过程的 3 个形参。

2）%let 语句为定义宏变量，引用宏参数传递的变量值。

3）数据步中数据集名 sj&v_name 为 sj 与宏变量名 v_name 传递值组合的名字。

4）%readfile（D:\jx\xs,dat,msjxsdat）为宏调用语句，传递实参文件路径与文件名 D:\jx\xs 给宏变量 v_dir。dat 为实参格式传递给 v_type。msjxsdat 为数据集名传递给 v_name。

【提示】 此程序具有通用性，读取其他类型的文件只需要传递相应的参数就可以。

3.3.8 表格数据处理

表格数据也是经常见到的一类数据，在做调查问卷、经济数据分析、医药分析、病人跟踪调查时经常用到。

表格数据处理中经常用到 Hash。Hash 为快速存取数据提供高效的、方便的方法，基于查找索引。

定义 Hash 语法格式：

DECLARE object variable<(<argument_tag-1: value-1<, ...argument_tag-n: value-n>>)>;

【语法解读】

DECLARE：声明 Hash 的关键字，必写项。

Object：指定对象，只能是 Hash 或 Hiter。Hash 指定一个哈希对象，哈希对象提供了一种对数据的快速存储和检索，哈希对象存储和检索数据按照关键主键查找；Hiter 指定一个哈希迭代器对象，哈希迭代器对象通过正向或反向键顺序帮助检索哈希对象的数据。

Variable：指定 Hash 或 Hiter 的名字，也就是给 Hash 定义名字；

argument_tag：指定定义 Hash 或 Hiter 信息。Hash 和 Hiter 的区别在 argument_tag 参数上。

5 个有效的哈希对象参数标签如下。

1）dataset: 'dataset_name <(datasetoption)>'：指定装载的数据集名字加载到哈希对象。数

据集名称必须括在单引号或双引号内。宏变量必须括在双引号内。使用数据集的选择时，声明一个哈希对象中的数据参数标签。执行以下操作：

① 命名变量。

② 选择一个子集数据加工。

③ 对数据集有条件选择，如删除或保留数据集中的变量。

④ where 过滤条件过滤数据。以一组数据加载到一个哈希对象或一个输出数据集，在一个指定的输出方法调用。

⑤ 指定一个密码数据集。

为便于理解，下面举一个简单的声明 Hash 语法的例子。

DECLARE hash h_stu (dataset: 'stu (where = (class = 10))');

【语法解读】

h_stu：为声明的 Hash 对象名。

dataset：声明指定数据集标签，stu 为数据集名。

where=语句为过滤条件，取数据集符合条件的数据。

2）duplicate: 'option'：对重复键设置。通过这个标签参数的定义，确定装载数据集到哈希对象时是否忽略重复键数据，默认是存储第一个关键值数据，其他的忽略。“option”取的值如下。

'replace' | 'r'：存储最后重复键记录。

'error' | 'e': 如果发现重复键记录报告错误的日志。

3）hashexp: n，指定 Hash 表格空间。

4）ordered: 'option' 按照索引指定排序。

5）suminc: 'variable-name'：保持一个汇总数哈希对象键。

【例 3.40】 经济分析调查数据，如表 3-3～表 3-5 所示。

表 3-3 消费调查

调查地区	公司名	调查时间	年收入/万元
北京	沃尔玛	2011-02-29	30 000
北京	沃尔玛	2011-03-15	28 000
北京	家乐福	2011-02-29	24 000
北京	家乐福	2011-03-15	29 000
天津	沃尔玛	2011-02-29	28 000
天津	沃尔玛	2011-03-15	27 000
天津	家乐福	2011-02-29	21 000
天津	家乐福	2011-03-15	19 000

表 3-4 北京地区

调查地区	调查时间	收益率（%）
北京	2011-02-29	30
北京	2011-03-15	28

表 3-5　天津地区

调查地区	调查时间	收益率（百分比）
天津	2011-02-29	27
天津	2011-03-15	22

对以上数据进行合并生成新数据表，如表 3-6 所示。

表 3-6　合并后表格数据显示样式

调查地区	公司名	调查时间	年收入/万元	收益率（%）
北京	沃尔玛	2011-02-29	30 000	30
北京	沃尔玛	2011-03-15	28 000	28
北京	家乐福	2011-02-29	24 000	27
北京	家乐福	2011-03-15	29 000	22
天津	沃尔玛	2011-02-29	28 000	
天津	沃尔玛	2011-03-15	27 000	
天津	家乐福	2011-02-29	21 000	
天津	家乐福	2011-03-15	19 000	

```
*表 3-3  消费调查数据处理;
data    inve_incom;
Length    time    $10.;
input    area $    company    $    time    $    anav_income    $;
cards;
beijing      沃尔玛      2011-02-29 30000
beijing      沃尔玛      2011-03-15 28000
beijing      家乐福      2011-02-29 24000
beijing      家乐福      2011-03-15 29000
tianjin      沃尔玛      2011-02-29 28000
tianjin      沃尔玛      2011-03-15 27000
tianjin      家乐福      2011-02-29 21000
tianjin      家乐福      2011-03-15 19000
;

run;
*表 3-4 beijing 地区数据处理;
data    beijing;
length    time    $10.;
input    area $    time $    rate $;
cards;
beijing      2011-02-29 30
beijing      2011-03-15 28
;
run;
*表 3-4 tianjin 地区数据处理;
```

```
data    tianjin;
length    time    $10.;
input area    $    time    $    rate    $;
cards;
tianjin        2011-02-29 27
tianjin        2011-03-15 22
;
run;
/*通过 hash 处理 3 个数据集，根据 area、time 关键字 hash 数据*/
data    invest_analy;
       if    0 then    set    beijing;/*如果是一开始，读入数据集 beijing*/
         if    _n_=1    then    do; /*只是在第一次执行声明 hash*/
       declare    hash    hax;
end;

set    inve_incom;
  hax=_new_    hash(dataset:area,ordered:'ascending');
  hax.defineKey ('area','time');
  hax.definedata(all:'yes');
  hax.defineDone ();
  record=hax.find();
run;
/*hash 不能对中文关键字处理，转换为拼音，下面的程序把拼音表示的中文地区转换为中文*/
data    invest_look(drop=record);
set    invest_analy ;
select    (area);     /*对 id 字段选择查询*/
when    ('beijing')     area='北京';     /*条件若成立执行 area='beijing'*/
when    ('tianjin')     area='天津';     /*条件若成立执行 area='tianjin'*/
otherwise    area='其他省份';       /*上面条件都不成立执行 area='其他省份'*/
end;
run;

proc    print    data=invest_look;
var    area    company    time    anav_income rate;/*按 var 语句指定的顺序显示输出*/
run;
```

【程序解读】

1）对表格数据先通过数据步处理，生成 SAS 能够识别的数据集。

2）通过 hash 处理数据集，“declare hash hax;”语句声明哈希，哈希名为 hax。

3）hax=_new_hash(dataset:area,ordered:'ascending');对哈希定义，数据集标签为 area，ordered 指定按升序排列。

4）hax.defineKey ('area','time');定义查找匹配关键字主键为 area 和 time 两个字段。

5）hax.definedata(all:'yes');定义数据域为所有。

6）hax.defineDone ();哈希定义结束语句。

7）record=hax.find();哈希查找语句。

3.3.9　二次数据处理

实际开发中通常数据处理并不是一次性就能达到满足业务需求的数据，需要根据实际需要进行二次数据处理。

SAS 系统不能直接对外部数据文件或其他关系数据库数据所具有的数据进行分析，需要先通过 SAS 程序转换数据到 SAS 系统，从而生成 SAS 系统能够识别的数据集，然后再根据需要对已经生成的数据集进行二次开发处理，生成满足业务需求的数据，从而便于前端的展现和分析。

【例 3.41】 以金融统计数据为例，通过 SAS 程序分析全国银行间同业拆借市场交易期限分类统计，如表 3-7 所示。

表 3-7　全国银行间同业拆借市场交易数据统计

	1 天		7 天		14 天		20 天	
	交易量	加权平均利率（%）	交易量	加权平均利率（%）	交易量	加权平均利率（%）	交易量	加权平均利率（%）
2011 年累计	281 600.18		487 009.81		9 765,21		3 456.85	
2012.01	12 345.17	4.11	5 343.23	4.13	663.22	5.56	987.12	5.19
2012.02	12 341.07	4.31	2 344.81	4.26	965.22	4.51	784.12	7.11
2012.03	12 342.23	4.21	1 345.71	4.28	762.22	4.59	885.12	5.14
2012.04	12 343.59	4.34	2 348.89	4.23	861.22	5.32	687.12	6.12
2012.05	12 346.32	4.21	2 335.36	4.29	665.22	5.31	987.12	7.13

```
*处理原始数据;
data  interbank;
input  trad_vol  warate  @@;
cards;
281600.18 . 487009.81 . 9765.21 . 3456.85 .
12345.17 4.11 5343.23 4.13 663.22 5.56 987.12 5.19
12341.07 4.31 2344.81 4.26 965.22 4.51 784.12 7.11
12342.23 4.21 1345.71 4.28 762.22 4.59 885.12 5.14
12343.59 4.34 2348.89 4.23 861.22 5.32 687.12 6.12
12346.32 4.21 2335.36 4.29 665.22 5.31 987.12 7.13

;
run;
*二次数据处理，生成按年对应的数据;
data  clbank;
set  interbank;
if _n_<5  then  year=2011;
   else  if _n_<9  then  year=201201;
    else  if  _n_<13 t hen  year=201202;
      else  if  _n_<17  then  year=201203;
        else  if  _n_<21  then  year=201204;
         else  year=201205;
```

```
        run;
    *调用 means 过程对经济数据按年分析，;
        proc   means   data=clbank   ;
            by   year;
    run;
```

【程序解读】

1）首先通过数据步处理原始数据，生成数据集 interbank。

2）对数据集 interbank 二次处理，根据需要生成对应按年的数据。每一年对应 4 条记录，if 语句判断生成按年对应的数据。

3）调用 means 过程对数据集 clbank 进行均值分析，按年分析。

第 4 章　过程步基础与案例

4.1　过程步基础

SAS 过程步（proc step）是 SAS 系统的另一个核心步，对生成的数据集进行分析和处理，挖掘数据信息。相对数据步而言过程步比较固定，功能被封装，用户只需要知道每一个过程实现什么功能，用到时根据功能需求调用相应的过程即可。过程步就像一个模板，基本上不需要修改什么。过程步没有语句的伸缩性，很多都是固定的。

4.1.1　过程步功能与定义

1．过程步功能

SAS 系统通过 SAS 数据步生成和管理数据，调用相应的过程进行分析、生成报表、绘图。SAS 过程步是对生成的数据集进行处理和分析，是 SAS 内部已经编译好的过程，用户直接根据业务需求，调用 SAS 内部过程并根据业务需求对所调用过程选项设置进行分析处理、作图和报表，然后根据调用过程输出的信息写出分析报告，做总结性评价。

2．过程步定义

理解过程步定义是掌握过程步的关键，过程步与数据步一样，也有自己的语法规范，严格执行语法规范才能正确地调用过程步。过程步说明如表 4-1 所示。

过程步语法格式：PROC 过程名 <DATA=数据集名>　<选项>;
　　　　　　　　　　　过程语句 <参数选项>;
　　　　　　　　　RUN;

表 4-1　过程步说明

过程步操作语句	功　能
PROC	过程步开始标志关键字，告诉 SAS 系统此处为过程步
过程名	SAS 系统内部编译好的过程，直接写过程名
DATA=	指定过程步处理的数据集，省略取当前最新 work 数据集
选项	控制过程的选项，如 double|d：，在观测记录之间写一空行，每个过程有自己的选项，同时还有通用选项
过程语句	过程内部分析数据集用到的语句
参数选项	过程语句具有的选项，如分析变量选项
RUN	告诉过程步到此处结束，也可以省略此语句，在下一个过程步或数据步的开始处结束

4.1.2　过程步应用

过程步实际业界应用中经常用到的是过程步过程语句。过程步基本语句是对所调用过程进行辅助分析，通过对过程步过程语句的选择，使分析和处理数据集的功能更强大。过程步

语句分为通用语句和专用语句，通用语句适合每一个过程，专用语句只适合本过程步所调用的过程具有的语句。使用时要注意区分，哪些是通用语句，哪些是某个过程专用语句。

过程步通用基本语句如表 4-2 所示。

表 4-2　过程步通用基本语句

过 程 语 句	功　　能
VAR	指定分析变量，多个变量用空格分隔
BY	指定一个或多个分组变量对数据集分组，数据集要先排序
CLASS	指定一个或多个分类变量，不需要事先对数据集排序
MODEL	建模中指定模型的因变量和自变量以及相关模型选项
FREQ	指定用于分析变量的频数变量
WEIGHT	指定权数变量，以获得该变量占百分比的权重
ID	指定观测标识，输出会去掉 OBS 选项标识
OUTPUT	对过程分析结果输出到新数据集
WHERE	条件选项语句
TITLE	输出文件加入标题信息
FOOTNOTE	输出文件加入脚注信息
OPTIONS	通过此语句设置，改变 SAS 系统的默认设置

【提示】 过程步中的过程语句与数据步中的语句不同，数据步中的语句不能用到过程步中；过程步中过程语句以某一个关键字开始，如 BY、VAR、CLASS、WEIGHT、FREQ、MODEL 等。

【例 4.1】 对一个班级的学生成绩进行分析，求出一个班级数学、英语和语文 3 门课程的总成绩、最高分和最低分。

```
data   score;
input   id   name   $   class   math   english   chinese;
cards;
1001  高名  1 89.2 78.3 89
1002  仲小海  1 76 99 78
1003  刘海洋  1 88 56 66
1004  杨小帅  1 99 89 98
1005  赵小红  1 87 86 83
1006  马西瑞  1 89 58 43
;
run;
/*过程步调用 means 过程求出一个班级数学、英语和语文的总成绩、最高分和最低分*/
proc   means   data=score   sum   max   min ;
var   math   English   chinese;
run;
```

程序执行后的结果如图 4-1 所示。

输出 - （无标题）

SAS 系统　　2012年05月15日 星

MEANS PROCEDURE

变量	总和	最大值	最小值
math	528.2000000	99.0000000	76.0000000
english	466.3000000	99.0000000	56.0000000
chinese	457.0000000	98.0000000	43.0000000

图 4-1　score 数据集分析结果

【程序解读】

1）过程步调用 means 过程，data=选项指定对数据集 score 分析。

2）sum、max、min 三个选项对 var 语句指定的分析变量 math、English、Chinese 三个变量分别求出一个班级的总成绩、最高分和最低分。

4.2　常见过程步应用

SAS 系统内部过程比较多，本节通过常见过程的讲解告诉读者学习过程的方法。

4.2.1　print过程

print 过程属于打印输出过程，对要查看的数据集可以通过调用 print 过程进行打印输出到 SAS 系统的输出窗口。一般经常用 print 过程查看生成的数据集。

print 过程语法格式：

```
PROC　PRINT　<DATA=数据集名>　<选项>;
  BY 变量名 1 <变量名 2>…;
  FOOTNOTE<n> <'footnote'>;
  FORMAT 变量名 1 输出格式名;
  ID 变量名 1 <变量名 2>…;
  LABEL 变量名='标签名'…;
  PAGEBY 变量名;
  SUM 变量名 1 <变量名 2>…;
  SUMBY 变量名;
  TITLE<n> <'标题内容'>;
  VAR 变量名 1 <变量名 2>…;
WHERE 条件语句;
RUN;
```

【语法解读】

1）<DATA=数据集名>：指定数据集名，省略时为当前临时库最新数据集。

2）<选项>：可以取的选项如 DOUBLE|D 指定数据记录之间加入一空行，LABEL 指定打印输出标签，NOOBS 指定不显示观测序号。

【例 4.2】 打印输出学生数据集中各科成绩都大于 80 分的学生信息。

```
 data    score;
   input   id name   $ class   math   English   chinese;
cards;
1001  高名  1 89.2 78.3 89
1002  仲小海  1 76 99 78
1003  刘海洋  1 88 56 66
1004  杨小帅  1 99 89 98
1005  赵小红  1 87 86 83
1006  马西瑞  1 89 58 43
;
run;
/*过程步调用 print 过程打印输出各科成绩都大于 80 分的学生信息*/
proc   print   data=score noobs;
       where   math>80 and   english>80   and   chinese>80;
/*where 条件语句，and 语句表示且，三者都符合才取出数据打印输出*/
   title '各科成绩大于 80 分的学生信息';
/*title 语句指定标题*/
run;
```

【程序解读】

1）data=score：指定打印数据集 score。

2）noobs：指定不打印输出序号。

3）where 条件语句指定满足各科成绩都大于 80 的学生信息打印输出到输出窗口。

4）title 语句指定打印标题信息。

程序执行后输出窗口显示打印结果如图 4-2 所示。

输出 － （无标题）

各科成绩大于80分的学生信息

2012年05月1

id	name	class	math	english	chinese
1004	杨小帅	1	99	89	98
1005	赵小红	1	87	86	83

图 4-2 成绩大于 80 分的学生信息

4.2.2 means过程

means 过程是 SAS 提供的一个数据汇总统计过程，提供单个或多个变量的简单描述统计分析。与 univariate 过程比较，means 过程倾向于描述已经明确的样本所在总体符合正态分布的变量，不提供百分位数，可以提供 95%的可信区间，对全体观测或分组观测进行描述性统计，如基于距计算描述统计、分位数估算（包括中位数）、识别极值、T 测试。

means 过程语法格式：

```
PROC MEANS <DATA=数据集名> <选项> <统计关键量>;
  VAR 变量名 1 <变量名 2>…;
  BY  变量名 1 <变量名 2>…;
CLASS  变量名 1 <变量名 2>…;
  FREQ <变量名>;
WEIGHT <变量名>;
  ID  变量名 1 <变量名 2>…;
OUTPUT <OUT=数据集名>  关键字=<新变量名列>;
RUN;
```

【语法解读】

1）means：所调用的过程名。

2）<DATA=数据集名>：指定要分析的数据集名，默认为当前最新数据集。

3）<选项>：常用选项，如 noprint 禁止统计报告输出到 OUTPUT 窗口；maxdec=n 指定列表输出的最大小数位数，默认为 2。

4）<统计关键量>：常用统计关键字，如 max（求最大值）、min（求最小值）、mean（求均值）、sum（求和）等，具体还有很多，用到时查 SAS 帮助。

【例 4.3】 某奶厂用自动装奶机装奶，在装奶机正常工作时，每瓶奶净重 450g，某日随机抽取了 10 瓶成品，称重分别为 452、436、447、439、445、460、442、456、447、440，请问这时的装奶机是否正常工作？

分析：若装奶机正常，当前已经装奶的全部产品净重与 450g 无统计意义差异。将分析变量的值减去正常值，得到一组新样本，做均值为零的 T 检验。

```
data   tcheck;
input   milck   @@;
milck=milck-450;/*与正常值的差*/
cards;
452 436 447 439 445 460 442 456 447 440
;
run;
/*调用 means 过程进行 T 检验*/
proc   means   data=tcheck   t   prt;   /*T 检验*/
run;
```

程序执行后结果显示如图 4-3 所示。

【程序解读】

1）数据步 milck=milck−450；语句求出实际值与正常值的偏差。

2）调用 means 过程，进行 T 检验。

结论：T 检验中 p=0.1737>0.05，假设成立，因此装奶机正常。

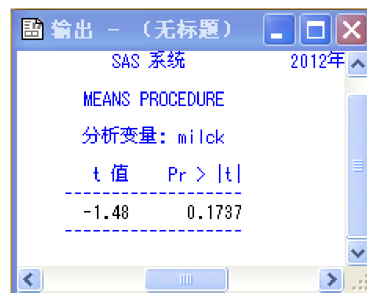


图 4-3　装奶机 means 分析 T 检验

4.2.3　copy过程

copy 过程可以复制一个逻辑库下的所有数据集到另一个逻辑库下或者从一个文件复制到另一个文件，也可以有选择地复制逻辑库下的数据集。

copy 过程语法格式：

PROC COPY in=源逻辑库名 out=目标逻辑库名<memtype=成员类型> <move <alter=修改密码>><index=YES|NO>;

SELECT 成员名</<memtype=成员类型> <alter=修改密码>;

EXCLUDE 成员名</<memtype=成员类型>;

RUN;

【语法解读】

1）copy：所调用的过程名。

2）in=：选项指定源逻辑库名。

3）out=：选项指定目标逻辑库名。

4）memtype=成员类型：可选项，规定 SAS 数据库中一个或几个被复制或移动成员的类型，如 memtype=（data catalog）。

5）move：选择指定在成功复制到输出库 out=指定的逻辑库后，从源逻辑库（in=源逻辑库）中删除已经复制的成员。

6）SELECT：选择复制的数据集名

7）EXCLUDE：排除不需要复制的数据集名。

【提示】 SELECT 语句与 EXCLUDE 语句不能同时出现，功能相反。

【例 4.4】 复制 jx 逻辑库中的数据集 stu 和 stu2 到逻辑库 mb 中，并从 sy 逻辑库删除 stu 和 stu2 数据集。

```
libname   jx 'd:\jx';            /*创建源逻辑库*/
libname   sy 'd:\test';          /*创建目标逻辑库*/
data   jx.stu;
   input   id   name   $   class;
cards;
1001  高兴  1
1002  你好  1
;
run;
data   jx.stu2;
   input   id   name $   class;
cards;
1001  董好  2
1002  刘小名  2
;
run;
/*调用 copy 过程复制数据集到目标逻辑库*/
proc   copy   in=jx   out=sy move;
/*in=指定源逻辑库名，out=目标逻辑库名，move 选择指定复制成功从源逻辑库删除复制的数据集*/
```

```
    select    stu    stu2;
/*选择复制的数据集名*/
run;
```

【程序解读】

1）调用 copy 过程，in=指定源逻辑库名，out=目标逻辑库名。

2）move 选项指定后，数据集复制成功后会从 jx 逻辑库中删除。

3）select 语句指定选择 stu 和 stu2 两个数据集复制到目标逻辑库 sy。

4.2.4 SQL过程

SQL（Structured Query Language）过程是 SAS 内部的一个过程，相当于数据库里的存储过程，功能都封装在此过程里，只需要通过过程步调用就可以运用此过程。SQL 过程也是和外部数据库进行交互的一个过程，同时此过程可以对数据集进行处理，一些语句和关系数据库中的 SQL 语言用法基本相同。

通过 SQL 过程可以对数据集或关系数据库的表进行查询、修改，实现创建表、删除数据、插入数据和更新数据等功能。

SQL 过程可以说实现了关系数据库的结构化查询功能，体现了 SAS 对大型数据库管理系统（如 ORACLE、DB2、Sybase）通用的 SQL 语言支持。SQL 过程说明，如表 4-3 所示。

语法格式：
```
PROC SQL    <选项>;
        数据操纵语句;
QUIT;
```

表 4-3　SQL 过程说明

SQL 过程操作语句	功　　能
PROC	过程步开始处关键字，调用过程语句
SQL	SAS 内部编译好的过程名，执行 SQL 语言
选项	SQL 过程具有的设置参数，如 NOPRINT
数据操纵语句	关系数据库结构化语句或对数据集操纵的语句，如查询
QUIT	告诉 SAS 系统 SQL 过程步结束

【提示】 SQL 过程的结束语句标志为“QUIT”语句。

SQL 过程可以对 SAS 数据集操作，也可以对关系数据库中的表操作，功能强大，继承了关系数据库中的 SQL 语言。

SQL 过程中的选项是控制 SQL 过程输出显示设置的应用。SQL 过程选项及功能如表 4-4 所示。

表 4-4　SQL 过程选项及功能

选　　项	功　　能
PRINT	打印 SQL 过程的输出到输出窗口，默认选项
NOPRINT	不打印 SQL 过程的输出到输出窗口
NUMBER	对输出观测记录指定行号（ROW）

（续）

选　　项	功　　能
NONUMBER	对输出观测记录不指定行号（ROW），默认选项
INOBS=	指定输入的观测记录行数，取正整数
OUTOBS=	指定输出的观测记录行数，取正整数
LOOPS=	指定 SQL 过程内循环的最大次数
DOUBLE	指定在各行插入一空行
NODOUBLE	指定在各行不插入一空行，默认选项
DQUOTE=	对双引号中的内容而言，取值为 ANSI 表示为变量，取值为 SAS 表示为字符，默认值取 SAS

SQL 数据操纵语句是对数据集或关系数据库中的表进行处理的语句，同时具有关系数据库 SQL 语言的操纵语句。实际开发和运用中并不是单个 SQL 语句的应用，而是多个 SQL 语句的组合应用共同完成一个任务。常用数据操纵语句如表 4-5 所示。

表 4-5　常用数据操纵语句

常用数据操纵语句	功　　能
CREATE TABLE	创建关系数据库中的表或 SAS 数据集
CREATE INDEX	创建索引，对大表可以提高查询速度
CREATE VIEW	创建视图
SELECT	查询关系数据库中的表或 SAS 数据集的信息
DELETE	删除关系数据库中的表或 SAS 数据集中的行记录
ALTER TABLE	对表或数据集的列变量进行修改，加入列或删除列
DROP	删除表、视图或索引
INSERT　INTO	插入观测记录
UPDATE	更新表或数据集中的对应列值
DESCRIBE	显示表或视图的定义信息
CONNECT TO	建立与关系数据库的连接
DISCONNECT FROM	断开与关系数据库的连接

【例 4.5】 查询学生成绩数据集 stu_score，语文成绩大于 60 分小于 80 分的显示及格，大于等于 80 分的显示优秀，其他显示不及格；数学成绩大于 60 分小于 80 分的显示及格，大于等于 80 分的显示优秀，其他显示不及格；英语成绩大于 60 分小于 80 分的显示及格，大于等于 80 分的显示优秀，其他显示不及格。

```
data   stu_score;
  Input   id   name   $   chinese   math   english;
cards;
1001  高兴  58 78 90
1002  马小名  78 38 88
1003  刘小华  89 90 87
1004  董下小  60 80 52
1005  杨小名  38 45 51
1006  张与信  99 89 87
```

```
;
run;
/*调用 SQL 过程，通过 case when 语句实现*/
proc  sql;
  select   name as 姓名, (case  when  chinese>=80  then '优秀'
                     when  chinese>=60  then  '及格'
                      else  '不及格'  end)  as 语文,
                  (case  when  math>=80  then  '优秀'
                     When  math>=60  then  '及格'
                      else  '不及格'  end)  as 数学,
                 (case  when  english>=80  then  '优秀'
                     when  english>=60  then  '及格'
                      else  '不及格'  end)  as 英语
     from  stu_score;
quit;
```

程序执行后的结果如图 4-4 所示。

图 4-4 学生成绩显示

【程序解读】

调用 SQL 过程，select 语句中的 case when 语句对变量根据条件处理相应的数据值。

【例 4.6】 北京地区举办金融业羽毛球比赛，比赛成绩如表 4-6 所示。

表 4-6 比赛成绩

队　名	日　期	结果（胜或负）
金羽	2012-05-01	胜
金羽	2012-05-01	胜
金羽	2012-05-01	负
金羽	2012-05-1	负
金羽	2012-05-02	胜
飞翔	2012-05-02	负
飞翔	2012-05-02	胜
飞翔	2012-05-02	负

请求出每个对当前胜与负的次数，显示形式如表 4-7 所示。

表 4-7 需求结果

队　名	日　期	胜	负
金羽	2012-05-01	2	2
金羽	2012-05-02	1	0
飞翔	2012-05-02	1	2

程序如下：

```
/*羽毛球比赛结果数据集 ym*/
data   ym;
length   sj   $10.;
input   dm   $   sj   $   jg   $;
cards;
金羽   2012-05-01      胜
金羽 2012-05-01 胜
金羽 2012-05-01 负
金羽 2012-05-01 负
金羽 2012-05-02 胜
飞翔 2012-05-02 负
飞翔 2012-05-02 胜
飞翔 2012-05-02 负
;
run;
/*调用 SQL 过程*/
proc   sql;
select   dm  as  队名,sj  as  日期, sum(case   when  jg='胜'   then  1   else  0   end)   '胜',sum(case  when
jg='负' then 1 else 0   end)   '负'   from ym
group   by   dm , sj            /*指定按 dm 和 sj 列分组*/
order   by   dm   desc, sj   asc;   /*指定按 dm 列降序，按 sj 列升序*/
quit;
```

【程序解读】

调用 SQL 过程，通过 SUM 函数和 case when 语句的联合应用实现对胜负求和功能。

执行程序后输出窗口显示结果如图 4-5 所示。

输出 － （无标题）

SAS 系统　　2012年05月17

队名	日期	胜	负
金羽	2012-05-01	2	2
金羽	2012-05-02	1	0
飞翔	2012-05-02	1	2

图 4-5　比赛结果显示

4.2.5　report过程

SAS 系统中的 report 过程是制作报表的工具，将 print、means 和 tabulate 过程的特点与 DATA 步报告写法的特点结合起来组合成了一个强大的生成报表的工具。

语法格式：　PROC　REPORT <报表选项>;

常用报表设置语句;

RUN;

1．报表选项

报表选项是进行报表开发时设置报表属性以及制定数据集的选项，属于可选项，如表 4-8

所示。

表 4-8　常用报表选项及说明

报 表 选 项	说　　明
data=	指定数据集
headline	指定在抬头行和第一个细目行之间输出一行下画线
headskip	指定在抬头行和第一个细目行之间输出一空白行

2．常用报表设置语句

常用报表设置语句是 report 过程根据需要生成各类报表需要的报表设置语句。常用报表设置语句及说明如表 4-9 所示。

表 4-9　常用报表设置语句及说明

常用报表设置语句	说　　明
title	指定报表标题信息
title2	指定报表副标题信息
break	控制一个分组变量或次序变量的值改变时的中断作用
by	根据 by 指定变量分组生成单独报表
column	指定报表列显示顺序
compute	定义计算变量，并执行行计算，或生成中断行
define	定义变量并对变量指定类型
freq	求出指定变量的频数
rbreak	对报表过程产生一个概括统计
weight	求出变量的权重

【例 4.7】 房价经济指数数据，环比与同比、定基数据生成报表。

```
libname  re  'd:\jx';
data  re.house;
input  city  $  hb_index  same_index  def_index;
cards;
北京 99.9 100.3 102.4
天津 99.8 100.2 103.1
秦皇岛 99.8 100.5 106.3
石家庄 99.7 101.3 107.7
包头 99.8 100.0 103.9
太原 100.0 100.9 101.7
;
run;
proc  print  data=re.house;
run;
/*调用 report 过程生成报表*/
proc  report  data=re.house  headline  headskip;
title '六个大中城市住宅销售价格指数 (2012 年 2 月)';
```

```
title2 '单月城市销售价格';
column   city   hb_index   same_index   def_index   dif;
define   city /order   format=$6.   width=6 '城市';
define   hb_index/display   format=5.1   width=5 '环比';
define   same_index/display   format=5.1   width=5 '同比';
define   def_index/display   format=5.1   width=5 '定基';
define   dif/computed   format=5.1   width=5 '差比';
compute   dif ;
dif=same_index-hb_index;
endcomp;
run;
```

【程序解读】

1）headline：指定表头画线。

2）headskip：指定一空行。

3）title：指定主标题。

4）title2：指定副标题。

5）column：指定报表显示列。

6）define：对报表显示列定义属性。

7）compute：指定计算列，其结束语句为 endcomp。

4.2.6 freq过程

计数资料的统计推断，常用的是 freq 过程。freq 过程主要用于两个目的：一是描述分析，产生频数表和列联表，可简单地描述数据；二是统计推断，产生各种统计量，可分析变量间的关系。freq 过程可以创建单向频数表、双向和 n 向交叉表。它还可以计算关联测度和一致性测度，并依据分层变量排列输出。

语法格式：　PROC FREQ　<选项>;

　　BY　变量 1 <变量 2>…<变量 n>;

　　EXACT　统计选项</计算选项>;

　　OUTPUT <out=数据集名>　选项:

　　TABLES requests </选项>;

　　TEST　选项;

　　WEIGHT　变量名</选项>;

RUN;

【语法解读】

FREQ：过程步所调用的过程名。

BY：指定分组变量，通过单独分组后对每个组单独分析。

EXACT：指定精确测试。

OUTPUT：指定输出数据集。

tables：tables 语句是 freq 过程中非常重要的一条语句。在一个 PROC FREQ 过程中，可以有任意多个 tables 语句。如果没有 tables 语句，FREQ 对数据集中的每个变量都生成一个单

向频数表；如果 tables 语句没有任何说明选项（options），freq 对 tables 语句中规定的变量的每个水平将计算频数、累计频数、占总频数的百分数及累计百分数。tables 语句中，用 request（s）指定制表要求。可以是用*连接起来的一个变量或多个变量，一个 tables 语句中可以给出任意多个制表要求。在 TABLES 语句的斜杠（/）后面可以使用的常用选项如下。

CHISQ：进行卡方检验（Chi-Square Test）。

Fisher：对大于 2×2 的表进行 Fisher 的精确检验。

ALPHA=p：确定置信区间的水平是 100（1-p）%的置信区间，默认 p=0.05。

ALL：对所有由 CHISQ、MEASURES 和 CMH 选项给出的检验和度量。

其他选项有 NOCOL、NOROW、NOPERCENT、NOFREQ。

TEST：指定相关的计算关联测度和一致性测度。

WEIGHT：指定一个权重变量。

【例 4.8】 根据班级分组，求出每个班级的频数。

```
 libname   jx 'd:\jx';
data   jx.class;
input   class   name   $   math   english   chiness;
cards;
1  高洪  89 78 99
1  王红与  78 45 32
1  李小心  74 72 78
2  马小名  87 98 86
2  刘小红  56 82 83
2  董西幂  88 91 92
3  杨小青  78 56 87
3  张峰余  56 78 23
3  赵晓霞  90 98 96
;
run;
/*调用 freq 过程对数据集分析*/
proc   freq   data=jx.class;
   by   class;/*按 class 班级分组，求出每个班级的频数*/
run;
```

【程序解读】

1）data=：选项指定要求频数的数据集。

2）by：指定按 class 班级分组变量。

【例 4.9】 投掷硬币的实验，投掷实验 100 次，出现的正面和反面次数如表 4-10 所示。

表 4-10　投掷硬币实验

组　名	正　面	反　面
甲组	49	50
乙组	52	48
丙组	47	53

```
data    yb;
   input    group    $    row    colm    fre;
cards;
甲组  1 1 49
甲组  1 2 50
乙组  2 1 52
乙组  2 2 48
丙组  3 1 47
丙组  3 2 53
;
run;
proc    print ;
run;
/*调用 freq 过程，进行频数分析*/
proc    freq    data=yb;
  tables    group*row * colm;/*指定制表列*/
  weight    fre;/*指定分析权重为 fre*/
run;
```

【程序解读】

1）tables 语句指定指标列表，首先按 group 分组后绘制 row 和 colm 组成的表。

2）weight 语句指定分析权重为 fre。

4.2.7 summary过程

summary 过程主要用来对数值变量计算单个变量的基本统计量，使用语句与 means 过程类似。默认时 summary 过程步打印输出计算结果。如果想打印输出计算结果，必须指定 print 选项。

```
语法格式：  PROC    SUMMARY    <选项><统计关键量>;
                    VAR 变量名 1 <变量名 2>…;
                    BY  变量名 1 <变量名 2>…;
CLASS  变量名 1 <变量名 2>…;
 FREQ <变量名>;
WEIGHT <变量名>;
                   ID  变量名 1 <变量名 2>…;
OUTPUT <OUT=数据集名> <选项>;
RUN;
```

【语法解读】

1）<选项>：常用选项如下。

data=：指定分析的数据集。

maxdec=：输出小数点位数。

missimg ：要求遗漏数据作为 class 变量下的一个分组处理。

print：指定打印输出到输出窗口，默认不打印输出结果到输出窗口。

nway：不对 class 变量中的统计量计算输出。

2）<统计关键量>：常用统计关键量如下。

n：有效观察值个数。

nmiss：遗漏数据个数。

mean：均值。

std：标准差。

min：最小值。

max：最大值。

range：极差。

sum：观测值总和。

var：样本方差 s2。

uss：总平方和。

css：总离差平方和。

stderr：均值的标准误。

cv：变异系数 s/mean。

T：t 检验 H0: 均值=0 prt---t 检验显著性。

3）其他语句参考 means 过程，功能类似。

【例 4.10】 取出数据集完全重复的数据。

```
data  information;
  input  name  $  class  math;
cards;
高小红 1 89
董与名 2 78
高小红 2 89
高小红 1 89
杨晓霞 1 99
高小红 1 89
董与名 2 78
;
run;
/*取出数据集完全相同的重复数据*/
proc  summary  data=information  nway;  /*nway 去掉不完全重复的统计记录*/
  class  name  class  math; /*分组变量*/
   output  out=cf (drop=_type_  where=(_freq_>1)); /*指定重复记录输出到数据集 cf 保存*/
run;
proc  print  data=cf;/*打印输出结果*/
run;
```

【程序解读】

1）nway：选项很重要，通过此选项去掉不完全重复的统计记录。

2）class：指定分组变量。

3）output：指定输出分析结果，out=指定输出到的数据集名。

4）where=(_freq_>1)：条件语句，取出重复记录条件。

【例 4.11】 学校学生体检数据的分析。

```
data   students;
 Input   group   age   height   weight   sex $ ;
Cards;
2 35 162 42 f
1 31 173 43 m
2 42 156 56 f
1 53 152 39 f
1 42 173 63 m
1 28 165 55 f
2 33 157 66 f
2 17 162 46 f
1 16 173 45 m
1 25 180 66 m
;
run;
/*调用  sort 过程首先对数据集排序*/
proc   sort   data=students;
 by   group ;   /*by  分组变量名串；对 by  变量的值排序（字母升序）*/
/*调用 Summary 过程*/
proc   Summary   data=students   mean   std   n   max   min   range   stderr   cv ;
/*指定统计关键量 mean   std n max min range stderr cv */
 var   age   height   weight; /*指定分析变量*/
class   sex;   /*指定分类变量，class  产生 3 类：不分 fm(以空白显示)及  f,m */
 by   group ;   /*by 指定分组变量，产生两类：1 和 2。以上两行语句分成 6  个水平组合进行统计*/
 output   out=stu_analy ;
proc   print   data=stu_analy;   /*打印分析结果*/
run;
```

程序执行后的结果如图 4-6 所示。

输出 － （无标题）

SAS 系统　　2012年05月23日 星期三 上午07时4

Obs	group	sex	_TYPE_	_FREQ_	_STAT_	age	height	weight
1	1		0	6	N	6.0000	6.000	6.0000
2	1		0	6	MIN	16.0000	152.000	39.0000
3	1		0	6	MAX	53.0000	180.000	66.0000
4	1		0	6	MEAN	32.5000	169.333	51.8333
5	1		0	6	STD	13.1263	9.730	11.1788
6	1	f	1	2	N	2.0000	2.000	2.0000
7	1	f	1	2	MIN	28.0000	152.000	39.0000
8	1	f	1	2	MAX	53.0000	165.000	55.0000
9	1	f	1	2	MEAN	40.5000	158.500	47.0000
10	1	f	1	2	STD	17.6777	9.192	11.3137
11	1	m	1	4	N	4.0000	4.000	4.0000
12	1	m	1	4	MIN	16.0000	173.000	43.0000
13	1	m	1	4	MAX	42.0000	180.000	66.0000
14	1	m	1	4	MEAN	28.5000	174.750	54.2500
15	1	m	1	4	STD	10.9087	3.500	11.9269
16	2		0	4	N	4.0000	4.000	4.0000
17	2		0	4	MIN	17.0000	156.000	42.0000
18	2		0	4	MAX	42.0000	162.000	66.0000
19	2		0	4	MEAN	31.7500	159.250	52.5000
20	2		0	4	STD	10.5633	3.202	10.7548
21	2	f	1	4	N	4.0000	4.000	4.0000
22	2	f	1	4	MIN	17.0000	156.000	42.0000
23	2	f	1	4	MAX	42.0000	162.000	66.0000
24	2	f	1	4	MEAN	31.7500	159.250	52.5000
25	2	f	1	4	STD	10.5633	3.202	10.7548

图 4-6　summary 过程对 students 的统计分析

【程序解读】

1）调用 summary 过程对数据集 students 进行分析。此程序指定了统计关键量：mean（求均值）、std（求标准差）、n（有效观察值个数）、max（最大值）、min（最小值）、range（极差）、stderr（均值的标准误）和 cv（变异系数）。

2）Var 语句指定分析变量为 age（年龄）、height（身高）和 weight（体重）。

3）Class 指定分类变量，根据 sex（性别）分类，产生 3 类，不分 fm（以空白显示），f，m。

4）By 指定分组变量，根据 group（分组）产生两类：1 和 2。每一类产生 3 类，共 6 个水平组合进行统计。

5）Output 指定输出结果到数据集。

6）调用 print 过程打印输出结果到输出窗口。

4.2.8 compare过程

compare 过程主要用来比较两个数据集的内容。

语法格式：
```
PROC COMPARE   <选项>;
   VAR 变量名 1 <变量名 2>…;
   BY  变量名 1 <变量名 2>…;
  ID   变量名 1 <变量名 2>…;
  WITH  变量名 1 <变量名 2>…;
RUN;
```

【语法解读】

1）调用 compare 过程。

2）<选项>：常用选项如下。

base=：指定基本数据集，省略时取最新数据集。

compare=（可以简写为 c 或 comp）：指定比较数据集。

Out=：指定输出数据集，把匹配变量的差值结果输出到指定数据集。

Outstats=：把匹配变量的概括统计量输出到指定的数据集。

Noprint：不打印输出到输出窗口。

Nomissing|nomiss：此选项判断缺失值与任何值相比较都相等。省略时，则一个缺失值仅与另一个同类的缺失值相等。

Brief：打印一个简短的摘要。

3）WITH 语句：两个数据集中对应变量的变量名不一致时用 WITH 语句列举他们的名字。WITH 语句必须与 VAR 语句一起使用，WITH 语句的第一个变量对应 VAR 语句中的第一个变量，依次类推。

【例 4.12】 比较学生信息数据集 stu1 和 stu2 的差异。

```
 data   stu1;
   Input   group   age   height   weight   sex $ ;
cards;
2 35 162 42 f
1 31 173 43 m
```

```
2 42 156 56 f
1 53 152 39 f
;
run;
data   stu2;
  Input   group2   age   height   weight   sex   $ ;
Cards;
2 35 162 42 f
1 31 173 43 m
2 42 156 56 f

;
run;
proc   compare   base=stu1   c=stu2   out=cy   brief;/*brief 选项打印差异摘要信息*/
 Var   group;
  With   group2;
run;
```

程序执行后的结果如图 4-7 所示。

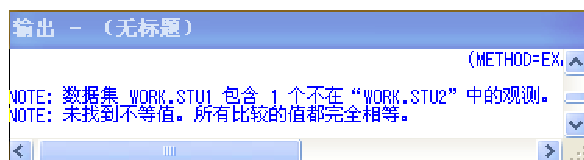


图 4-7　数据集比较摘要信息

【程序解读】

1）调用 compare 过程，base=stu1 指定基本数据集，c=stu2 指定比较数据集。

2）Out=指定差异输出结果到此数据集。

3）Brief：指定打印摘要信息，如果 4-7 所示的信息。

4）Var group：指定用于比较数据集中相匹配的变量列表，此语句只指定了 group 变量，比较 group 变量差异。

5）WITH 语句：两个数据集中对应变量的变量名不一致时，用 WITH 语句列举他们的名字。

4.2.9　datasets过程

datasets 过程主要用来对 SAS 逻辑库中的 SAS 文件进行列表、复制、换名、添加和删除等操作。此过程具有 append 过程、contents 过程和 copy 过程的功能。

Datasets 过程的主要功能如下：

1）对 SAS 逻辑库中的数据集从一个逻辑库复制到另一个逻辑库中。

2）对 SAS 文件进行重命名。

3）修复损坏的 SAS 文件。

4）删除 SAS 文件。

5）列出某一 SAS 逻辑库中所有的 SAS 文件。

6）列出某一 SAS 数据集的属性，如最近修改时间、数据是否压缩、数据是否索引等。

7）对 SAS 文件设置密码的操作。

8）向 SAS 数据集添加记录。

9）对 SAS 数据集的属性以及数据集内变量的属性进行修改。

10）创建或删除 SAS 数据集的索引。

11）创建并管理 SAS 数据集的核查文件。

12）创建或删除 SAS 数据集的完整性规则。

语法格式：
```
PROC DATASETS   <选项>;
        APPEND base=目标数据集  <data=追加数据集><force>;
        CHANGE  旧变量名=新变量名;
        CONTENTS <data=逻辑库.>表名>;
       COPY in=逻辑库名  out=逻辑库名<选项>;
       SELECT   选择数据集名;
       EXCLUDE  排除数据集名;
       DELETE  删除的数据集名;
       MODIFY  修改数据集名<选项>;
       FORMAT  变量 1  类型..;
       INDEX CREATE  索引名=<变量></<unique><nomiss>>;
       INDEX DELETE  索引名;
       INFORMAT  变量 1  类型..;
       LABEL   变量名 1='标签信息' …;
       RENAME  变量名=新变量名…;
       SAVE  数据集名</memtype=mtype>;

RUN;
```

【语法解读】

1）<选项>：常用选项如下。

Library=：指定逻辑库，给出要处理的 SAS 逻辑库。省略此选项时处理当前 work 逻辑库。

Memtype=：可以简写为 mtype 或 mt，规定一个或几个用于处理的文件类型。可以取值 all、catalog、data、access、view 和 program。

Kill：删除 SAS 逻辑库中的所有成员。

Force：对于 append 语句中，force 强制追加数据。另一个功能是有语法错误时强制退出。

Nolist：阻止列出在 SAS 的日志中正在处理的成员。

Pw=：给予数据库访问密码。

Read=：给出读数据库的访问密码。

Alter=：给出修改密码。

2）APPEND 语句：追加一个数据集到 base=语句指定的数据集中。

3）CHANGE 语句：改变数据集变量的名字。

4）CONTENTS 语句：查看指定的数据集信息。
5）COPY：复制 in=逻辑库的数据集到 out=指定的逻辑库中。
6）SELECT：选择要处理的数据集。
7）EXCLUDE：指定要排除的数据集。
8）DELETE：指定要删除的数据集。
9）MODIFY：指定要修改的数据集。
10）FORMAT：对变量指定输出格式。
11）INDEX：对索引通过 create 创建，通过 delete 删除。
12）INFORMAT：指定输入格式。
13）LABEL：对变量定义标签。
14）RENAME：对变量修改名字。
15）SAVE：指定要保存的数据集。

【例 4.13】 复制逻辑库 sashelp 中的数据集 air 、afmsg 和 adomsg 到逻辑库 jx。

```
libname   jx   'd:\jx\test';
proc   datasets   memtype=data;
    copy   in=sashelp   out=jx;   /*把 sashelp 逻辑库中的数据集复制到 jx 逻辑库*/
    select   air a fmsg   adomsg;   /*选择数据集 air afmsg adomsg 复制到逻辑库 jx*/
run;
```

【程序解读】

1）memtype=data：选项指定选择 data 类型数据。
2）copy 语句中 in=指定要复制的逻辑库，out=目标逻辑库。
3）select 指定选择数据集 air、 afmsg 和 adomsg 三个数据集，复制到逻辑库 jx。

【例 4.14】 删除逻辑库 jx 中的表 afmsg 和 adomsg。

```
libname    jx   'd:\jx\test';
proc   datasets   library=jx   memtype=data;
     delete   afmsg   adomsg;/*把 jx 逻辑库中的数据集 afmsg 和 adomsg 删除*/
quit;
```

【程序解读】

1）选项 library=jx 指定要操作的逻辑库对象。
2）Delete 语句指定要删除的数据集 afmsg 和 adomsg。

4.2.10 surveyselect抽样过程

在数据挖掘模型过程中，有时无法对所有的整体进行全面研究，通常将整体划分为训练集、验证集、测试集用于不同目的数据集，甚至在 K−折交叉验证中，需要把样本随机地划分为 K 份数据子集。SAS 通过 SURVEYSELECT 过程进行抽样，常用的抽样有单纯随机抽样（simple random sampling）、系统抽样（systematic sampling）、分层抽样（stratified sampling）、整群抽样（clustering sampling），SAS 系统通过 PROC SURVEYSELECT 过程实现对数据的抽样。

```
语法格式：PROC SURVEYSELECT;
                Strata  变量;
                id      变量;
          RUN;
```
【语法解读】

1）<选项>：常用选项如下。

data=：指定输入数据集。

Method=：指定抽样方法，如 srs、sys 和 urs。

out=：指定抽取样本存放的数据集。

n=：指定抽取数量。

samprate=：指定抽样比例。

seed=n：指定随机数。

2）strata 语句：指定分层变量。

3）id 语句：指定抽取样本所保留的源数据集变量。

【例 4.15】 从 1 万个样本中按随机数 100 抽取数据，选择简单随机抽样。

```
 data   sj ;
Do   i = 1   to 10000 ;
output ;
end ;
run;
/*随机抽样*/
proc   surveyselect
      data =sj
      method = srs
      n = 100
      out=cysj   seed =100;
run;
```

【程序解读】

1）method=srs：指定选择简单随机抽样。

2）n=100：指定抽样样本个数 100 个。

3）out=cysj：指定抽样数据保留到此数据集。

4）seed=100：指定抽样随机数为 100。

【例 4.16】 对例 4.15 改造，简单随机抽样，按 20%的比例抽取。

```
proc   surveyselect
      data =sj
      method = srs
      samprate=0.2
      out=cysj2   seed =100;
run;
```

【程序解读】

samprate=0.2：指定按 20%的比例抽取。

4.2.11 format过程

format 过程主要用来定义数值或符号文字的输出和输入格式，定义好的变量类型可以在数据步直接引用，以便于输出的 Output 更易于了解及阅读。在使用此程序时，注意这个程序只是“定义”并不会自动输出至 Output，必须用 PROC FORMAT 宣告的输出格式，配合以下两个程序步骤（Step）相对应的宣告值使用：

- 在 DATA 中的 INPUT 和 FORMAT 语句中应用定义的类型，可以在函数中应用。
- 在 PROC 中的 FORMAT 叙述语句中应用。

语法格式：PROC FORMAT <选项>;

value 变量名 1 <输出格式选项> 范围 1=’格式值’…;

invalue 变量 1<输入格式选项> 范围 1=’格式值’…;

picture 变量名 1 <输出格式选项> 范围 1=’格式值’…;

select ‘选择格式列表’;

exclude ‘选择格式列表’;

RUN;

【语法解读】

1）<选项>：常用选项如下。

① CNTLIN=：SAS 输入数据集名，指定一个包含可用于产生输出和输入格式的信息输入控制数据集。CNTLIN=提供的产生输入/输出格式的方式不需要在 value、picture 或 invalue 语句中规定所有信息。

② CNTLOUT=：SAS 输出数据集名，指定一个包含可用于产生输出和输入格式的信息输出控制数据集。不指定逻辑库时输出控制数据集存储到 work 逻辑库中。

③ LIBRARY=：指定逻辑库。

2）value：定义变量输出格式，定义好的变量可以在数据步引用。

3）invalue：定义变量输入格式，定义好的变量可以在数据步引用。

4）picture：给出定义的图示输出格式。

5）select：选择给出的输入/输出格式的目录子集。

6）exclude：排除给出的输入/输出格式的目录子集。

【例 4.17】 通过 format 过程实现卡编号转换。在数据步应用定义的变量类型。

```
/*Format 过程定义 card_type 类型*/
proc   format;
value   card_type
         1001='普卡'
         1002='金卡'
         1003='白金卡'
         1004='国航联名卡'
         1005='标白卡';
         RUN;
```

```
        DATA   card;
        Input   id   bh;
        cards;
        1001 345
        1002 456
        1003 870
        1004 987
        1005 876
        ;
        run;
/*数据集 card 引用 format 过程定义的类型 card_type*/
data   trans_card;
          set   card;
            idbh=put(id,card_type.);/*引用 card_type 类型*/
        run;
/*打印输出结果，查看数据集*/
proc   print   data=trans_card;
run;
```

程序执行后，通过打印过程输出显示数据集信息，如图 4-8 所示。

输出 - （无标题）

SAS 系统 201

Obs	id	bh	idbh
1	1001	345	普卡
2	1002	456	金卡
3	1003	870	白金卡
4	1004	987	国航联名卡
5	1005	876	标白卡

图 4-8　类型转换

【程序解读】

1）调用 foramt 过程，通过 value 语句定义变量输出格式 card_type，并指定变量值的范围。

2）数据步对数据集 card 处理，idbh=put(id,card_type.);引用 format 过程定义的类型 card_type，把 id 号转换为 card_type 输出格式的变量。

【例 4.18】 学生成绩显示应用，小于 60 分显示不及格，大于等于 60 分小于 80 分为良好，大于等于 80 分为优秀。

```
proc   format;
value   scor_desc   0-<60='不及格'
                    60-<80='良好'
                    other='优秀';
run;
data   stu_score;
               input   id   name   $   score;
               cards;
               1001  高宏  58
               1002  马小名  60
               1003  刘晓华  80
               1004  董青青  90
               1005  杨峰  78
               ;
run;
```

/*打印过程调用 format 过程定义的类型，通过 foramt 语句引用类型*/
proc print data=stu_score;
format score scor_desc.;/*format 语句引用 format 过程定义的类型*/
run;

程序执行后，输出窗口显示如图 4-9 所示。

【程序解读】

1）foramt 过程定义变量类型 scor_desc，具体的值范围直接列出。

2）print 打印过程中通过 format 语句对数据集变量 score 定义为 scor_desc.类型作为输出格式。

输出 - （无标题）

SAS 系统 2012

Obs	id	name	score
1	1001	高宏	不及格
2	1002	马小名	良好
3	1003	刘晓华	优秀
4	1004	董青青	优秀
5	1005	杨峰	良好

图 4-9 stu_score 信息输出显示

4.2.12 sort过程

sort 过程将数据集中某一个变量或几个变量的按升序或降序重新排列，并把结果存储到输出数据集中。如果不另外指定输出数据集，则覆盖输入数据集。在 data 数据步和 proc 过程步的某些操作中，当需要用到 by 语句时，一般都需要源数据集按照 by 语句中的变量事先排序，这里就需要用到 sort 过程。sort 过程先检查输入数据集的排序信息，特别是 sortedby=选项，如果输入数据集提示已经按照 by 变量进行过排序，或者 sort 过程检测到数据集中记录的顺序按照 by 变量本来就是有序的，则 sort 过程不进行排序，直接将输入数据集复制到输出数据集中。另外，如果输入数据集在 by 变量上已经创建索引，则也不进行排序，因为排序之后会破坏原来的索引。除此之外，sort 过程才会进行排序。如果用户强制 sort 过程进行排序就需要用到 force 选项了。

语法格式：PROC sort <用户主机环境><选项>;
BY <DESCENDING> 变量 1 <...<DESCENDING> 变量 n>;
RUN;

【语法解读】

1）<用户主机环境>：指除 ASCII 或 EDCDIC 之外的排序序列。

2）<选项>。

① 常用数据集选项如下。

data=：指定要排序的数据集，默认时使用最近创建的数据集。

out=：指定排序后输出的数据集名字，默认时表示排序后覆盖源数据集。

② 常用排序序列选项如下。

sortseq=：指定排序的序列，这跟使用的操作系统有关，Windows/UNIX 都是 ASCII 编码。一般这个选项默认就可以，也可以直接在 sort 过程后面加上编码名称。

③ 常用修改排序次序选项如下。

reverse：使用由正常排序序列相反的排序序列对字符变量进行排序，可以被 by 语句中的 descending 选项取代，reverse 只能用于字符变量。

equals|noequals：规定输出数据集中具有相同 by 变量的那些记录的次序，equals 选项是保持在输入数据集中原来的相对次序，而 noequals 选项则没有这一限制。

④ 常用删除重复记录选项如下。

Noduprecs：删除重复的记录，发生在排序后，将完全相同的记录删除。

nodupkey：删除重复的 by 变量记录，发生在排序中，根据 by 指定的变量判断，sort 过程读取输入数据集中的记录，在写入输出数据集时先比较 by 变量值，如果有重复则写入输出数据集。

⑤ 其他常用选项如下：

Datecopy：保留数据集创建或修改的日期。

Force：强制排序选项，不管输入的数据集是否已经排序或有索引，都进行重新排序。

by：by 语句指定排序变量顺序，默认情况下是按照变量进行升序排列（ascending），降序则要用 descending 指明。特别是这两个关键字应该写在变量的前面，而其他语言可能相反，如 SQL 将排序关键字放在变量之后。

【例 4.19】 去除学生数据集完全重复的记录，输出到数据集 stu_order。

```
data    stu;
        Input    id    name    $    score;
         cards;
         1001  高宏  58
         1002  马小名  60
         1003  刘晓华  80
         1004  董青青  90
         1005  杨峰  78
         1006  杨峰  98
             1003  刘晓华  80
         ;
run;
        /*sort 过程通过 noduprecs  去除完全相同的记录，结果输出到数据集 stu_order*/
proc    sort    data=stu    out=stu_order    noduprecs ;
            by    name;
srun;
```

【程序解读】

1）out=：指定排序后数据集输出到 stu_order 数据集，不覆盖原数据集。

2）noduprecs：去除完全重复的记录。

3）by：by 语句指定按 name 升序排列。

【例 4.20】 对例 4.19 进行改造，去除学生数据集中姓名相同的记录，并按姓名降序排列，输出到数据集 st_order。

```
data    stu;
        Input    id    name    $    score;
         cards;
         1001  高宏  58
         1002  马小名  60
         1003  刘晓华  80
         1004  董青青  90
         1005  杨峰  78
```

```
        1006 杨峰 98
          1003 刘晓华 80
        ;
run;
       /*Sort 过程通过 noduprecs 去除完全相同的记录，结果输出到数据集 st_order*/
proc   sort   data=stu   out=st_order   noduprecs ;
            by   descending   name;
run;
```

【程序解读】

by：by 语句中 descending 指定按变量 name 降序排列。

【提示】 上面的 Sort 过程把 nodumprecs 换成选项 nodupkey 是去除根据 by 语句指定的 name 变量相同的记录，而不是去除完全相同的记录。

4.3　经济指数指标分析案例

经济指数指标是研究经济运行情况的指标，是每一个国家为维护经济稳定和长远发展必须关注的经济点。目前中国经济处于发展期，农产品价格指数是研究国内农产品价格趋势的分析数据。农产品价格波动会影响广大居民的生活水平，需要国家对农产品价格进行合理的定价调控，长期监控其价格趋势，有助于稳定居民生活水平，使其处于合理的价位。

【例 4.21】 调查 2012 年农产品价格指数，分析农产品价格在 2012 年前 5 个月农产品总体价格趋势。蔬菜价格调查数据，如表 4-11 所示。

表 4-11　蔬菜价格调查数据

年份/年	月份/月	蔬 菜 价 格
2012	1	2.3
2012	2	3.1
2012	3	2.9
2012	4	3.2
2012	5	2.8

```
/*农产品经济指数统计，agri_Proc 为上涨率*/
data   ecnomic;
input   y_moth   $   agri_pro;
cards;
201201 2.3
201202 3.1
201203 2.9
201204 3.2
201205 2.8
;
run;
data   analy;
```

```
set    ecnomic;
cz=agri_pro-lag(agri_pro);    /*求出每条记录与上月的差值*/
run;
proc    means    data=analy;
var    cz;
run;
```

程序执行后的输出结果如图 4-10 和图 4-11 所示。

输出 － （无标题）

SAS 系统　　2012年05月2

Obs	年月	农业产品价格比	比率差值
1	201201	2.3	.
2	201202	3.1	0.8
3	201203	2.9	-0.2
4	201204	3.2	0.3
5	201205	2.8	-0.4

图 4-10　比率差值信息

输出 － （无标题）

SAS 系统　　2012年05月29日 星期二 上

MEANS PROCEDURE

分析变量：cz 比率差值

N	均值	标准差	最小值	最大值
4	0.1250000	0.5377422	-0.4000000	0.8000000

图 4-11　means 过程分析比率差值

【程序解读】

1）cz=agri_pro-lag(agri_pro);：语句通过 lag()函数取上一条记录变量的值，然后通过减法运算求比率差值。

2）调用 means 过程，通过 var 语句对 cz（比率差值）进行分析。

【结论】 通过分析可以看出农产品价格在 2012 年前 5 个月总体处于上涨趋势，需要通过国家行政手段给予调控，防止通胀压力过大。

第 5 章　函数基础与案例

5.1　函数基础

每一个编程语言都有很多函数（function），SAS 语言也有很多函数。SAS 中 SAS 函数是 SAS 内部已经编译好的功能程序的封装，是把数学公式或其他子功能通过函数来实现。每一个函数都体现了某一个功能，根据实际需求合理而有效地调用这些函数可以实现强大的编程能力，提高程序的可读性和执行效率。

5.1.1　函数功能与常用函数

单从概念上讲，在数学领域函数是一种关系，这种关系使一个集合里的每一个元素对应另一个（可能相同的）集合里的唯一元素。

1．函数功能

函数功能是编程语言中实现满足某种需求的程序封装，相当于用户看到的电视画面，其实显示画面的功能都在电视里面，用户看不到，也不需要知道，只要知道用哪个按钮来控制电视就可以，这个控制电视的按钮就相当于函数名。函数功能的具体体现需要结合数据步和宏语言的应用来实现，实际的业界开发中经常在数据步和宏语言中调用大量的 SAS 内部函数，来帮助解决业务需求的问题。

1）一般函数书写形式。

一般书写形式就是按实际需求把每一个参数都写上，书写形式如下：

函数名（参数 1,参数 2,…）

2）简写形式。

简写形式是对同一个函数实现相同的功能，参数的书写形式不同的表示方法称为简写形式。

① 函数名(OF 参数 1-参数 n)

【函数解读】 参数 1 与参数 n 之间用“-”连接。

② 函数名（OF 参数 1 参数 2…参数 n）

【函数解读】 多个参数用空格分割。

2．常用函数

SAS9.2 具有很多已经编译好的函数，在 SAS9.2 之前是不支持自定义函数的，到了 SAS9.2 版本，SAS9.2 实现了一个大跳跃，用户可以自定义函数。所谓的自定义函数就是用户根据需要自己写程序，实现解决实际问题的某种功能。SAS9.2 版本系列目前的函数分为 25 种，函数处理功能有算术函数、字符函数、截取函数、概率与密度函数、数学函数和日期、时间函数、样本统计函数和分位数函数等。

① 字符处理函数

实际业界开发中经常接触到字符，如字母 what、中文字符：杨小红等都称为字符。根据开发需求，有时需要提取字符串中的某个字母或中文字符串中的某几个汉字。处理字符的函数称为字符处理函数，这类函数只能处理字符，不能处理数值。常用字符处理函数如表 5-1 所示。

表 5-1　常用字符处理函数

字 符 函 数	功 能 说 明
SUBSTR	字符串截取与替换
CAT	连接字符串
CATS	连接字符串前去掉字符串前面和后面的空格
CATX	去掉前后的空格后，在连接的两字符串中间插入分隔符
SCAN	查找到的特殊字符处开始分隔字符串
TRIM	过滤字符或字符串尾部空格
INDEX	查找一个字符串中第一次出现的某个字符的位置
INDEXC	查找字符串中想要找的字符中的任意一个首先出现的字符，并返回该字符位置，多个字符之间用逗号隔开
INDEXW	查找字符串中想要找的一个“单词”，并返回该单词位置，可设定单词之间的分隔符，属于完全匹配查找
LENGTH	求字符串的长度
COMPRESS	从字符串中移除指定的字符或字符子串
TRANSLATE	字符串替换
UPCASE	将指定的字符或字符串小写转换为大写
LOWCASE	将指定的字符或字符串大写转换为小写
TRANWRD	对指定字符串替换
REPEAT	对指定字符串根据指定次数重复书写
LEFT	左对齐字符
REVERSE	对指定的字符串按相反顺序输出

1）SUBSTR 函数。

语法格式：SUBSTR(变量名, 位置<,长度>)<=替换字符串>

【语法解读】

变量：要处理的字符串变量。

位置：指明开始查找字符串的位置，数值型。

长度：指明替代字符串的长度，数值型。

替换字符串：指定用来替换的字符。

功能：实现字符串的替换和截取功能。

【提示】 此函数对变量截取与替换，实际开发中对读取的外部文件需要根据业务需求截取原变量中的一部分，给新变量赋值。

【例 5.1】 SUBSTR 函数截取功能应用，读取外部数据文件，存储在 d:\jx 目录中的 cus_in.txt 文件，从身份证号中取出每一个人的出生年月日。

```
%let    path= D:\jx\cus_in;    /*定义外部文件路径*/
%let    type=.txt;
%let    wj= "&path&type";
libname    jx    'd:\jx';    /*定义逻辑库*/
data    jx.cust_in;    /*数据集存储到指定逻辑库*/
        Infile      &wj ;
                        input         id          5.
                                           cus_name       $8.
                                           sf_id          :$18. /*身份证号*/
                                           address    :$18.
                              ;
         length    v_age $8.; /*定义了一个业务需求变量 v_age，代表客户的出生年月日*/
                        v_age=substr(sf_id,7,8);
/*从身份证号码中取出客户的出生年月日*/
run;
```

【程序解读】

程序中通过 LENGTH 语句定义了新变量 v_age。此变量的值为 substr(sfz,7,8)函数从身份证号码第 7 位截取，取 8 位的值，这是身份证号的年月日的开始位置与长度。

【例 5.2】 SUBSTR 函数替换功能应用，对例 5.1 进行改造。

```
%let    path= D:\jx\cus_in;    /*定义外部文件路径*/
%let    type=.txt;
%let    wj= "&path&type";
libname    jx    'd:\jx';    /*定义逻辑库*/
data    jx.cust_inth;    /*数据集存储到指定逻辑库*/
        Infile    &wj ;
                        input         id          $5.
                                           cus_name       $8.
                                           sf_id          :$18. /*身份证号*/
                                           address    :$18.
                              ;
        /*id 前三位替换为 101*/
        substr(id,1,3)='101';

run;
```

【程序解读】

通过 SUBSTR 函数对 id 字符变量截取出 3 位，对取到的 3 位用字符串‘101’替换。

2）CAT、CATS 和 CATX 三个函数。

CAT、CATS 和 CATX 函数都是实现字符串连接的函数。

CAT 与 CATS 函数的语法格式相同。

语法格式：CAT（项 1, … , < 项 n>）

【语法解读】

项 1：可以取常值、变量或表述字符串。

< 项 n>：取第 n 个常值、变量或表述字符串。

功能：实现字符串的连接。

【提示】 CATS 函数去掉连接串中间空格，而 CAT 不去掉中间空格。

CATX 函数语法格式： CATX（分隔符，项 1，… , < 项 n>）

【语法解读】

分隔符：指定连接的两字符串中间插入分隔符。

功能：实现字符串的连接。

【例 5.3】 CAT、CATS 和 CATX 函数实现字符串连接的差异，对比学习其差异。

```
data   _null_;
length   v_constrcat   $20.   v_constrcats   $20.   v_constrcatx   $20.;
v_str1= 'MISS ';   /*此变量 MISS 后有个空格*/
v_str2= 'YANG';
v_constrcat= cat(v_str1,v_str2);
v_constrcats= cats(v_str1,v_str2);
v_constrcatx=catx(',',v_str1,v_str2);/*连接字符之间插入逗号*/
put   v_constrcat=
          v_constrcats=
           v_constrcatx=;
run;
```

程序执行后日志窗口显示信息如图 5-1 所示。

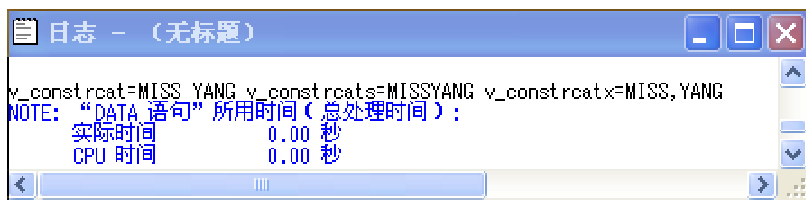


图 5-1 CAT、CATS 和 CATX 三个函数实现字符串连接对比输出

【程序解读】

① v_constrcat= cat(v_str1,v_str2)：实现字符串 v_st1 和 v_str2 两个字符串连接，中间空格保留。

② v_constrcats= cats(v_str1,v_str2)：实现字符串 v_st1 和 v_str2 两个字符串连接，去掉中间空格。

③ v_constrcatx=catx(',',v_str1,v_str2)：实现字符串 v_st1 和 v_str2 两个字符串连接，并在两字符串之间插入逗号分隔符号。

3）字符串拆分函数 SCAN。

SCAN 函数可以把以空格或标点符号隔开的第 n 个单词分隔开。SCAN 不同于 trim，trim 只是提取字符。

语法格式：SCAN(v_string, n, position, length <,delimiters>)

【语法解读】

v_String：为字符型常量、变量或表达式。

N：如果 n 是正的，从左到右扫描寻找；如果 n 是负的，从右到左扫描寻找。

Delimiters：指明字符串的分隔符。

功能：实现对特殊字符的查找，并从查找到的特殊字符处开始分隔字符串。

这个函数对处理特殊分隔符号的数据很有用，以特殊分隔符号为分界点，取出需要的数据。

【例 5.4】 SCAN 函数提取证件号应用，由于证件号码分为身份证、军官证和学生证，数据文件存储证件号这列数据格式为“B 证件号#证件类型标识”

说明：实际应用中经常用 SCAN 函数提取特殊分隔符号分隔的数据，以分隔符号为提取标志。

```
%let   path= D:\jx\zy_inf;   /*定义外部文件路径*/
%let   type=.txt;
%let   fil= "&path&type";
libname    jx 'd:\jx';   /*定义逻辑库*/
data   jx.zyinf;   /*数据集存储到指定逻辑库*/
        Infile   &fil ;
                    input          id                    $3.
                                   cust_name        $8.
                                   address              :$22. /*地址以#分隔*/
                                   zy_type          :$16.
                    ;
length   provice $10.;/*取省份或直辖市*/
                    provice=scan(address,1,'X#');
/*参数为正值，1 从左面扫描到右面，把以 X 和#号分隔的省或直辖市取出来*/
length dqu $10.;
   dqu=scan(address,-1,'X#');
   /*参数为负值时，-1 表示从右边开始扫描，把以#号分隔的地区取出来*/
run;
/*打印输出结果到输出窗口*/
proc   print   data=jx.zyinf;
run;
```

程序执行后输出结果显示如图 5-2 所示。

输出 - （无标题）

SAS 系统　　　2012年05月30日 星期三 上

Obs	id	cust_name	address	zy_type	provice	dqu
1	010	杨小红	X北京市#海淀区	教师	北京市	海淀区
2	022	刘晓乐	X山东省#青岛市	公务员	山东省	青岛市
3	010	马小滨	X天津市#滨海区	科技人员	天津市	滨海区

图 5-2　SCAN 函数应用取分隔符分隔字符串

【程序解读】

① provice=scan(address,1,'X#')：参数 1 为正值，从左到右扫描，查找以 X 和#分隔的字符串并取出来，赋值给 provice 变量。

② dqu=scan(address,−1,'#')：参数为负值时，−1 表示从右边开始扫描，把以#号分隔的地区取出来。

4）TRIM 函数。

TRIM 函数实现对常值、变量或表达式中的字符串过滤空格。

语法格式：TRIM（v_str）

【语法解读】v_str：为输入的常值、变量或表达式。

功能：过滤掉字符或字符串尾部空格。

【例 5.5】 拼接字符串时过滤 cust_name 尾部空格与字 address 变量连接。

```
%let   path= D:\jx\kg;   /*定义外部文件路径*/
%let   type=.txt;
%let   fil= "&path&type";
libname   jx 'd:\jx';   /*定义逻辑库*/
data   jx.kg;   /*数据集存储到指定逻辑库*/
       Infile   &fil dlm='|'   dsd   missover   ;
                    input          id                :$3.
                                    cust_name        :$8.
                                    address           :$22. /*地址以#分隔*/
                                    zy_type          :$16.
                          ;
length lkg_address $50.;
lkg_address=trim(cust_name)||address;/*过滤掉 cust_name 尾部空格*/

run;
```

【程序解读】

数据步中语句“lkg_address=trim(cust_name)||address”中引入 trim 函数过滤 cust_name 变量的空格，然后通过串连接，赋值给变量 lkg_address。

5）INDEX、INDEXC 和 INDEXW 三个函数。

① INDEX 函数：返回在一个字符串中，某个特定字符或者字符串的位置。实际业界开发中一般和 SUBSTR 函数一起使用，先查找位置，然后截取字符串。

语法格式：INDEX(source,find_v)

【语法解读】

Varable：指定的字符串。

find_v：指定要查找的字符。

功能：寻找在一个字符串中第一次出现的某个字符的位置。

② INDEXC 函数：查找指定字符串中要找的字符中的任意一个首先出现的字符，并返回该字符位置，多个字符之间用逗号隔开。

语法格式：INDEXC(source,excerpt−1<,…, excerpt−n>)

【语法解读】

source：可以为字符常量、变量或表达式字符。

excerpt-1<,…,excerpt-n>：指定要查找的字符常量、变量或表达式字符。

功能：查找指定字符串中要找的字符中的任意一个首先出现的字符，并返回该字符位置。

③ INDEXW 函数：查找指定字符串中要找的一个“单词或字符子串”，并返回该单词或字符子串位置，可设定单词或字符子串之间的分隔符，属于完全匹配查找。

语法格式：INDEXW(source, excerpt<,delimiters>)

【语法解读】

source：可以为字符常量、变量或表达式字符。

excerpt：指定要查找的字符常量、变量或表达式字符。

<,delimiters>：可选项，默认空格分隔，指定分隔字符常量、变量或表达式字符。

功能：查找字符串中要找的一个“单词或字符子串”，并返回该单词或字符子串位置，属于完全匹配查找。

【例 5.6】 INDEX、INDEXC 和 INDEXW 三个函数对字符串实现查找功能，对比学习其区别。

```
%let   path= D:\jx\zy_inf;   /*定义外部文件路径*/
%let   type=.txt;
%let   fil= "&path&type";
libname   jx 'd:\jx';   /*定义逻辑库*/
data   jx.zyindexcw;   /*数据集存储到指定逻辑库*/
       Infile   &fil ;
                  input          id                 $3.
                                  cust_name         $8.
                                  address           :$22. /*地址以#分隔*/
                                  zy_type           :$16.
                  ;
Length   diqu   $10.;
   diqu=substr(address,index(address,'#')+1,6);
   /*index 函数查找出‘#’位置，加 1 为要取子串开始位置，长度为 6*/
  length   v_indexc   $10.;
  v_indexc=substr(address,indexc(address,'#'),1);
  /*indexc 函数查找出#号第一次出现的位置，substr 函数取出#号*/
  v_indexw=indexw(address,'X 北京市#海淀区');
  /*indexc 函数查找出#号第一次出现的位置*/
RUN;
/*打印输出结果到输出窗口*/
proc   print   data=jx.zyindexcw;
run;
```

程序执行后的输出结果显示如图 5-3 所示。

输出 － （无标题）

SAS 系统　　2012年05月30日 星期三 上午07时

Obs	id	cust_name	address	zy_type	diqu	v_indexc	v_indexw
1	010	杨小红	X北京市#海淀区	教师	海淀区	#	1
2	022	刘晓乐	X山东省#青岛市	公务员	青岛市	#	0
3	010	马小滨	X天津市#滨海区	科技人员	滨海区	#	0

图 5-3　INDEX、INDEXC 和 INDEXW 三个函数实现查找字符

【程序解读】

① diqu=substr(address,index(address,'#')+1,6)：index 函数查找出‘#’位置，加 1 为要取子串的开始位置，长度为 6。

② v_indexc=substr(address,indexc(address,'#'),1)：indexc 函数查找出#号第一次出现的位置，substr 函数取出#号。

③ v_indexw=indexw(address,'X 北京市#海淀区')：indexc 函数查找出"X 北京市#海淀区"字符串，存在返回 1，不存在返回 0。

6）LENGTH 函数。

语法格式：LENGTH (varble)

【语法解读】 Varble：指定的变量。

功能：返回字符串变量的长度。

【提示】 实际业界开发中如果需要计算变量长度，可以考虑用 LENGTH 函数。对于需要从变量右边截取 n 个字符，可以通过 LENGTH 函数计算出总长度，然后通过 LENGTH (varble)−n+1 计算出开始位置。

【例 5.7】 LENGTH 函数计算字符串长度，与 SUBSTR 函数联合应用。

```
%let   path= D:\jx\kg;   /*定义外部文件路径*/
%let   type=.txt;
%let   fil= "&path&type";
libname   jx   'd:\jx';   /*定义逻辑库*/
data   jx.lid;   /*数据集存储到指定逻辑库*/
       Infile   &fil dlm='|'   dsd   missover   ;
                  input           id                  :$3.
                                  cust_name        :$8.
                                  address           :$22. /*地址以#分隔*/
                                  zy_type             :$16.
                      ;
Length    v_id $2.;
v_id=substr(id,length(id)−1,2);
   /*LENGTH 函数先求出字符串总长度，总长度减 1 为区号开始位置，然后通过外部 SUBSTR 函
数，从此位置开始取区号，取两位*/
run;
```

【程序解读】

v_id=substr(id,length(id)−1,2)：length 函数先求出变量 id 字符串总长度，总长度减 1 为区号开始位置，然后通过外部 SUBSTR 函数，从此位置开始取区号，取两位*/

7）COMPRESS 函数。

COMPRESS 函数从数据源移除指定的字符。这个函数可以去掉特殊字符等不需要的字符，相当于一个过滤。

语法格式：COMPRESS(<source><, chars><, modifiers>)

【语法解读】

source：数据源，指定一个字符串常量、变量或表达式。

chars：指定要移除的一个字符或字符子串常量、变量或表达式。默认情况下，在此列表中指定的内容被删除。如果在第三个参数 modifiers指定修饰符，那么只有在此名单中的字符保存在结果中，用此修改。

modifiers：指定修改查找到的内容方式，空格被忽略。作为修改参数，可以取的参数项值如下。

a或A：增加字母字符的字符列表。

c或C：增加了控制字符的字符列表。

d或D：增加了数字的字符列表。

f或者F：添加下画线字符和英文字母的字符列表。

g或G：增加了图形字符的字符列表。

h或H：增加了一个水平制表符的字符列表。

k或K：保留的字符。

l或L：增加了小写字母的字符列表。

n或N：增加了数字、下画线、英文字母和字符列表。

p或P：增加了标点符号的字符列表。

s或S：字符列表中添加空格字符（空白、水平制表符、垂直制表符、回车、换行、换页）。

u或U：增加了大写字母的字符列表。

w或W：增加了可打印字符的字符列表。

x或X：十六进制字符的字符列表。

【提示】 如果修改的是一个常数，用引号括起来，在单引号中指定多个常数，修饰符也可以表示为一个变量或表达式。

功能：对字符串移除。

【例 5.8】 COMPRESS 函数对 address 处理，移除指定的变量和保留指定的变量的应用。

```
%let   path= D:\jx\zy_inf;   /*定义外部文件路径*/
%let   type=.txt;
%let   fil= "&path&type";
libname   jx 'd:\jx';   /*定义逻辑库*/
data   jx.zycompress (keep=address v_addcompress1 v_addcompress2) ; /*数据集存储到指定逻辑库*/
       Infile   &fil ;
              input        id                $3.
```

```
                    cust_name           $8.
                    address            :$22. /*地址以#分隔*/
                    zy_type             :$16.
           ;
length    v_addcompress1 $30.;
          v_addcompress1 =compress(address,'X#'); /*移除 address 变量中的 X 和#字符*/
length    v_addcompress2 $10.;
   v_addcompress2 =compress(address,'X#','k'); /*k 参数表示保留 X 和#字符，其他字符从 address 变量中移除*/
RUN;
/*打印输出结果到输出窗口*/
proc   print   data=jx.zycompress ;
run;
```

程序执行后输出结果如图 5-4 所示。

输出 - （无标题）

SAS 系统　　2012年05月31日 星

Obs	address	v_addcompress1	v_addcompress2
1	X北京市#海淀区	北京市海淀区	X#
2	X山东省#青岛市	山东省青岛市	X#
3	X天津市#滨海区	天津市滨海区	X#

图 5-4　COMPRESS 函数处理变量应用

【程序解读】

① v_addcompress1 =compress(address,'X#')：通过 compress 函数移除 address 变量中包含的 X 和#字符。

② v_addcompress2 =compress(address,'X#','k')：通过 compress 函数，根据 k 参数，表示告诉 SAS 系统保留变量 address 中含有的 X 和#字符，其他字符从 address 变量中移除。

【提示】 通过对比可以发现，compress函数的第三个选项 modifiers的作用。读者可以对此案例进行改造，实验其他参数选项，在对比中学习其差异，才能提高对SAS的认识。

8）TRANSLATE 函数。

TRANSLATE 函数实现替换特定的字符功能，实际业界开发中经常用此函数处理缺失值的变量。由于包含缺失值的变量影响数据装载、统计分析和数据挖掘，往往缺失值都用一个常值替换。

语法格式：TRANSLATE(source,to−1,from−1<,…, to-n,from-n>)

【语法解读】

source：数据源，指定一个常量、变量或表达式。

to-1：指明要用的子串替换的字符串。

from-1：要从数据源替换的字符串。

【提示】 多个替换字符用逗号分隔。

【例 5.9】 TRANSLATE函数对address变量字符替换处理，把“# “号替换为“|”。

```
%let   path= D:\jx\zy_inf;   /*定义外部文件路径*/
%let   type=.txt;
%let   fil= "&path&type";
libname   jx 'd:\jx';   /*定义逻辑库*/
data   jx.zytranslate (keep=address v_addtranslate) ;   /*数据集存储到指定逻辑库*/
       Infile   &fil ;
                   input          id                $3.
                                  cust_name         $8.
                                  address           :$22. /*地址以#分隔*/
                                  zy_type           :$16.
                   ;
length v_addtranslate $30.;
       v_addtranslate =translate(address,'|','#'); /*translate 函数把变量 address 中的#字符用|替换*/
RUN;
/*打印输出结果到输出窗口*/
proc   print   data=jx.zytranslate;
run;
```

程序执行后输出结果如图 5-5 所示。

```
输出 - （无标题）
                         SAS 系统              2012年05月

     Obs        address              v_addtranslate

      1       北京市#海淀区         北京市|海淀区
      2       山东省#青岛市         山东省|青岛市
      3       天津市#滨海区         天津市|滨海区
```

图 5-5　TRANSLATE 函数应用

【程序解读】

v_addtranslate =translate(address,'|','#')：translate 函数把变量 address 中的#字符用|替换。

9）UPCASE 函数和 LOWCASE 函数。

UPCASE 函数和 LOWCASE 函数的功能相似，都是对字符或字符串大小写的转换。只是 UPCASE 函数将指定的字符或字符串小写转换为大写字符。LOWCASE 函数将指定的字符或字符串大写转换为小写字符。

语法格式：UPCASE(argument)

【语法解读】

Argument：要转换的字符 串常量、变量或表达式。

功能：把字符或字符串中的小写字母转换为大写。

语法格式：LOWCASE(argument)

【例 5.10】 UPCASE 函数把 level 变量对应的字符转换为大写。

```
data   car_level(keep=level v_level card_desc);
input   id :5.   level   $1.   type   $5.;
length   card_desc   $20.   v_level   $1.;
```

```
v_level=upcase(level);/*UPCASE 函数把变量 level 对应的字符转化为大写*/
select  (upcase(level));  /*select 语句应用函数*/
/*upcase 函数把变量 level 对应的字符串转化为大写，然后与 when 语句中的条件进行比较*/
   When  ('A')     card_desc=v_level||trim(type);
   When  ('B')      card_desc=v_level||trim(type);
   When  ('C')    card_desc=v_level||trim(type);
   otherwise       card_desc=v_level||trim(type); /*不符合以上条件的执行 otherwise 语句*/
end;
cards;
10001 a  金卡
10002 b  标白卡
10003 c  普卡
;
run;
/*查看数据集，打印输出到输出窗口*/
Proc  print  data=car_level noobs;
run;
```

程序执行后输出结果如图 5-6 所示。

【程序解读】

① v_level=upcase(level)：upcase 函数把变量 level 对应的字符转化为大写。

② select (upcase(level))：select 语句应用 UPCASE 函数把 level 对应的字符转化为大写，然后与 when 语句的字符进行比较。

【例 5.11】 对例 5.10 进行改造，LOWCASE 函数将 level 变量字符转换为小写。

```
data  car_levelxx(keep=level v_level card_desc);
input  id  :5.  level  $1.  type  $5.;
length  card_desc  $20.  v_level  $1.;
v_level=lowcase(level);/*LOWCASE 函数把变量 level 对应的字符串转化为小写*/
select  (lowcase(level));
/*LOWCASE 函数把变量 level 对应的字符串转化为小写，然后与 when 语句中的条件进行比较*/
   When  ('a')     card_desc=v_level||trim(type);
   When  ('b')      card_desc=v_level||trim(type);
   When  ('c')    card_desc=v_level||trim(type);
   otherwise       card_desc=v_level||trim(type); /*不符合以上条件的执行 otherwise 语句*/
end;
cards;
10001 A  金卡
10002 B  标白卡
10003 C  普卡
;
run;
/*查看数据集，打印输出到输出窗口*/
proc  print  data=car_levelxx  noobs;
run;
```

程序执行后输出结果如图 5-7 所示。

图 5-6 UPCASE 函数应用

图 5-7 LOWCASE 函数应用

【程序解读】

① v_level=lowcase(level)：LOWCASE 函数把变量 level 对应的字符转化为小写。

② select (lowcase (level))：select 语句应用 LOWCASE 函数把 level 对应的字符转化为小写，然后与 when 语句字符进行比较。

【提示】 对比例 5.11 和例 5.10 可以看到，UPCASE 函数和 LOWCASE 函数的功能相似。

10）TRANWRD 函数。

语法格式：TRANWRD(source,target,replacement)

【语法解读】

source：要处理的字符串。

target：指明要替换的子串。

replacement：替换的字符或子串。

功能：实现把字符串source中的 target选项 字符串替换为 replacement选项指定的字符或字符 串。

实际开发中经常用 TRANWRD 函数转换特殊分隔符号的外部文件，因为这类分隔符号标识分隔不唯一，SAS 系统无法区分，因此要把此分隔符号转换为 Windows 系统唯一分隔标识。

【例 5.12】 对 tsfg.txt 文件以|@|分隔的数据文件读取时需要先转换分隔，通过 tranwrd 转换分隔。

```
%let   path= D:\jx\tsfg;   /*定义外部文件路径*/
%let   type=.txt;
%let   fil= "&path&type";
libname   jx 'd:\jx';   /*定义逻辑库*/
data   jx.tsfg_trawrd;   /*数据集存储到指定逻辑库*/
       infile   &fil   dlm='08'x    dsd   missover    ;
input   @;/*行控制符，处理时不换行*/
/* 这样下面的转换函数 tranwrd(_infile_,'|@|','08'x)把'|@|'特殊字符分隔转换为'08'x 分隔的符号，
才能对外部文件处理 */
       _infile_=tranwrd(_INFILE_,'|@|','08'x);
                  input          id              :$5.
                                 cust_name       :$8.
                                 address         :$30.
                                 type            :$10.
```

```
                    ;
run;
```

【程序解读】

数据步中的语句“_infile_=tranwrd(_INFILE_,'|@|','08'x);”通过 TRANWRD 函数把分隔符号|@|转化为’08’x 分隔的文件。

11）REPEAT 函数。

REPEAT函数实现对指定的字符类型（如一个常量、变量或表达式）重复出现的次数。

语法格式：REPEAT(argument,n)

【语法解读】

argument：为输入字符或字符串，可以取一个常量、变量或表达式。

n：为自然数，取重复次数。

功能：把 argument字符或字符串重复n次。

【例 5.13】 对数据文件 cf.txt 中的缺失值用 5 个 8 替换。

```
%let   path= D:\jx\cf;   /*定义外部文件路径*/
%let   type=.txt;
%let   fil= "&path&type";
libname   jx 'd:\jx';   /*定义逻辑库*/
data   jx.cf_repeat(keep=idbh id);   /*数据集存储到指定逻辑库*/
     infile   &fil   dlm='08'x    dsd   missover    ;
input   @;/*行控制符，处理时不换行*/
/*  这样下面的转换函数 tranwrd(_infile_,'|@|','08'x)把'|@|'特殊字符分隔转换为'08'x 分隔的符号，
才能对外部文件处理  */
     _infile_=tranwrd(_INFILE_,'|@|','08'x);
              input            idbh          :$5.
                               cust_name       :$8.
                               address        :$30.
                               type           :$10.
                    ;
length   id   $5.;
      id= translate(idbh,repeat('8',5),' ');
  /*内嵌函数 REPEAT 把 8 重复 5 次输出，外部 TRANSLATE 函数处理缺失值，将数据文件中学
号 idbh 列生成数据集时的缺失值替换为 88888，赋值给 id 变量*/

run;
/*打印输出数据集*/
Proc   print   data=jx.cf_repeat;
run;
```

输出 － （无标题）

SAS 系统

Obs	idbh	id
1	01000	01000
2		88888
3	01011	01011

图 5-8　REPEAT 函数应用

程序执行后输出结果如图 5-8 所示。

【程序解读】

id= translate(idbh,repeat('8',5),' ')：内嵌函数 REPEAT 把 8 重复 5 次输出，外部 TRANSLATE 函数处理缺失值，将数据文件中学号 idbh 列生成数据集时的缺失值替换为 88888，赋值给 id

变量。

12）LEFT 函数。

LEFT 函数在字符串连接中经常用到，实现把左边空格滤掉。

语法格式：LEFT(argument)

【语法解读】 argument：为输入字符、字符串或表达式。

功能：把 argument字符或字符串左对齐，使左边空格去掉。

【例 5.14】 LEFT 函数在字符串连接中的应用。

```
*left 函数把字符串左对齐;
data   zf_left(keep=v_left no_left);
       v_s1='where are you from?';
        v_s2='   I am from china.';   /*语句左边有空格*/
       v_left=v_s1||left(v_s2);   /*LEFT 函数把字符串左对齐*/
        no_left=v_s1||v_s2;   /*没有 LEFT 函数字符串连接*/
run;
/*查看数据集，打印输出到输出窗口*/
proc   print   noobs;
run;
```

程序执行后输出结果如图 5-9 所示。

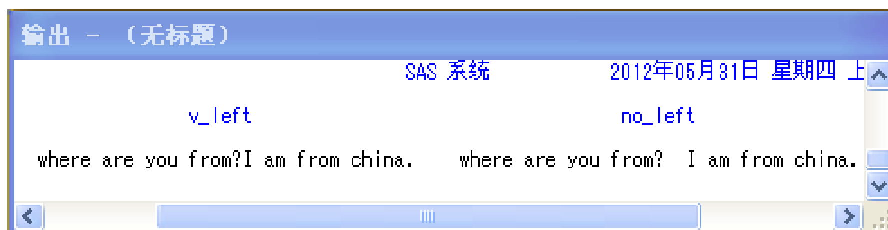


图 5-9　LEFT 函数应用

【程序解读】

对比 v_left 变量和 no_left 变量，可以看出 v_left=v_s1||left(v_s2)通过 left 函数过滤掉了左边的空格。

13）REVERSE 函数。

REVERSE 函数实现对指定字符串按相反的顺序输出。

语法：REVERSE(argument)

【语法解读】 argument：为输入字符或字符串或表达式。

功能：把 argument字符、字符串或表达式按相反的顺序输出。

【例 5.15】 对例 5.14 进行改造，REVERSE 函数转换输出顺序。

```
*reverse 函数把字符串按相反顺序转换字符串;
data   zf_reverse;
      v_s='where are you from?';
       v_reverse=reverse(v_s);/*按相反顺序转换字符串*/
```

```
run;
```

【程序解读】

v_reverse=reverse(v_s)：把 v_s 变量中的字符串按相反的顺序转换字符串。

（2）数学运算函数

数学运算是现实中经常遇到的，SAS 系统为满足实际需求，内部编译好了很多实现数学运算的函数，每一个数学函数实现一定的数学计算。常用数学运算函数如表 5-2 所示。

表 5-2 数学运算函数

数学运算函数	功 能 说 明
SUM	求出输入数值运算的和
MAX	求出一组数据中的最大值
MIN	求出一组数据中的最小值
ABS	取数值的绝对值
ROUND	对指定数值按指定的精度四舍五入
SQRT	求数值的平方根
CEIL	对数值取最小整数
FLOOR	对输入数值取最大整数
INT	对输入数值取整数部分，把小数部分去掉
LOG	对输入数值求自然对数
LOG2	求以 2 为底的输入值的对数
LOG10	求以 10 为底的输入值的对数
EXP	求以常值 e 为指数的 X 幂的值

1）SUM 函数。

SUM 函数是求给予的非缺失值数据的和运算函数。

语法格式：SUM(argument,argument, …)

【语法解读】 argument：为指定的数值常量、变量或表达式。

功能：求数值变量的和，实现加法运算。

【提示】 如果SUM函数指定一个参数选项，其函数返回值为 argument选项的值；如果指定两个或更多选项参数，其中有一个标准的缺失值点（.），其函数返回不是缺失值的和。

【例 5.16】 求一个班级每名学生 3 门课程的总成绩。

```
data  stu;
input  id  name  $  math  Chinese  English;
total=sum(math,chinese,english);  /*求每名学生 3 门课程的总成绩*/
cards;
101 高宏 78 80 90
102 马小名 89 87 78
103 董晓晴 65 98 76
;
run;
```

【程序解读】

total=sum(math,chinese,english)：sum 函数求出 math、Chinese 和 English 三门课程的总成绩，并赋值给变量 total。

2）MAX 函数和 MIN 函数。

MAX 函数求出一组数据的最大值，MIN 函数求出一组数据的最小值。

语法格式：MAX(argument-1,argument-2<,…,argument-n>)

【语法解读】 argument-1,argument-2<,…,argument-n>：为指定的多个数值类型参数选项，至少两个参数选项。

功能：求一组数值类型的最大值。

【例 5.17】 求出用户 4 个月刷卡交易的最高和最低交易额度。

```
    %let  path= D:\jx\trans;  /*定义外部文件路径*/
%let  type=.txt;
%let  fil= "&path&type";
libname  jx 'd:\jx';  /*定义逻辑库*/
data  jx.trans(keep=name month1 month2 month3 month4 tran_max tran_min);  /*数据集存储到指定逻辑库*/
      infile  &fil ;
      input  id        :$9.
             name      :$8.
             month1    :8.
             month2    :8.
             month3    :8.
         month4    :8.
             ;
length  tran_max  8.;
    tran_max=max(month1,month2,month3,month4);
 /*MAX 函数求出客户 4 个月中刷卡交易的最高额度*/
length   tran_min  8.;
    tran_min=min(month1,month2,month3,month4);
  /*MIN 函数求出客户 4 个月中刷卡交易的最低额度*/
run;
/*查看输出结果，打印输出到输出窗口*/
Proc  print  data=jx.trans;
run;
```

程序执行后输出结果如图 5-10 所示。

输出 - （无标题）

SAS 系统　　2012年05月31日 星期四 上午07

Obs	name	month1	month2	month3	month4	tran_max	tran_min
1	杨小红	1000	800	2000	4000	4000	800
2	刘小名	3000	4000	500	3800	4000	500
3	张小青	4500	3200	3600	300	4500	300

图 5-10　MAX 函数与 MIN 函数应用

【程序解读】

① tran_max=max(month1,month2,month3,month4)：通过 MAX 函数求出客户 4 个月中刷卡交易的最高额度。

② tran_min=min(month1,month2,month3,month4)：通过 MIN 函数求出客户四个月中刷卡交易的最低额度。

3）ABS 函数。

ABS（绝对值）函数主要用来对数值类型数据进行处理，对于一些实际意义不能为负的数据，通过 ABS 函数进行转换。

语法格式：ABS (argument)

【语法解读】 argument：为输入的数值参数。

功能：求变量 argument的绝对值。

【提示】 实际开发中人们总会犯错误，实际价格、交易额、成绩和年龄等不能为负值，在处理这类数据时通常应用 ABS 函数取绝对值，防止价格和年龄为负值的出现。

【例 5.18】 刷卡交易数据处理，数据文件 d:\jx\tran_card.txt。

```
 %let   path= D:\jx\trans_card;   /*定义外部文件路径*/
%let   type=.txt;
%let   fil= "&path&type";
libname   jx 'd:\jx';   /*定义逻辑库*/
data   jx.trans_abs;   /*数据集存储到指定逻辑库*/
      infile   &fil   dlm='|'     dsd   missover     ;
                  input                bh                      :$3.
                                         cust_name         :$8.
                                         tran_ed_1           :8.
                                         trans_address :$30.
                        ;
length   tran_ed   8.;
       tran_ed= abs(tran_ed_1);
  /*ABS 函数取交易绝对值，防止数值为负*/
drop   tran_ed_1;/*过渡变量 tran_ed_1 去掉*/
run;
```

【程序解读】

tran_ed= abs(tran_ed_1)：ABS 函数对源数据文件变量 ran_ed_1 取交易绝对值，防止数值为负。

4）ROUND 函数。

ROUND 函数实现按指定的精度取数值，对于需要指定精度处理的数据而言此函数可以满足需求。实际应用中小数点后面的数据有时没有实际意义，可以省略掉，也就是四舍五入。

语法格式：ROUND (argument <,rounding-unit>)

【语法解读】

argument：为输入的数值参数。

rounding-unit：为取的精度，不写此参数默认为 1。

功能：求 argument按 rounding-unit指定的精度四舍五入。

【例 5.19】 国家利率变动与存款贷款关系，换算利率按精度 0.001。

```
%let   path= D:\jx\rate;   /*定义外部文件路径*/
%let   type=.txt;
%let   fil= "&path&type";
libname   jx 'd:\jx';   /*定义逻辑库*/
data   jx.rate_round;   /*数据集存储到指定逻辑库*/
      infile   &fil   dlm='|'    dsd   missover    ;
            input             bh                :$3.
                              cust_name         :$8.
                              cust_money_1      :8.
                              rate              :5.4
            ;
length   custcq_money   8.;
      custcq_money= abs(cust_money_1);
  /*客户存款*/
length   cust_ydk   7.2;
  cust_ydk=custcq_money*(1+round(rate,0.001));
  /*ROUND 函数取利率精度，客户应得款为本金加利率所得*/
drop   tran_ed_1;
run;
```

【程序解读】

cust_ydk=custcq_money*(1+round(rate,0.001))：ROUND 函数取利率精度，客户应得款为本金加利率所得。

5）SQRT 函数、CEIL 函数、FLOOR 函数和 INT 函数。

这几个函数的基本功能相似，SQRT 函数求数值平方根，CEIL 函数对数值取最小整数，FLOOR 函数对输入数值取最大整数， INT 函数对输入数值取整数部分。

语法格式：SQRT(argument)

【语法解读】

argument：为输入的数值参数。

功能：求数值平方根。

语法格式：CEIL (argument)

【语法解读】

argument：为输入的数值参数。

功能：求比 argument数值大的最小整数，若x本身为整数，函数值为 argument。

语法格式：FLOOR (argument)

【语法解读】

argument：为输入的数值参数。

功能：求比 argument数值小的最大整数，若 argument本身为整数，函数值就为 argument。

语法格式：INT (argument)

【语法解读】

argument：为输入的数值参数。

功能：对输入数值取整数部分，把小数部分去掉。

【提示】 FLOOR 函数和 CEIL 函数功能相似， FLOOR 函数求比输入值小的最大整数，CEIL 函数求比输入值大的最小整数。对比其差异性来理解和运用这两个函数。

【例 5.20】 对变量 x 通过 SQRT 函数、CEIL 函数、FLOOR 函数和 INT 函数分别求平方根、求比 x 数值大的最小整数、求比输入值 x 小的最大整数和求 x 整数。

```
data   rate;
   input x 6.1;
      x_sqrt=sqrt(x);/*对 x 求平方根*/
      x_ceil=ceil(x); /*对 x 求比 x 数值大的最小整数*/
      x_floor=floor(x); /*对 x 求比输入值 x 小的最大整数*/
      x_int=int(x);/*对 x 求整数*/
cards;
4.0
8.1
8.2
9.5
;
run;
/*打印数据集到输出窗口，查看结果*/
proc   print   data=rate;
run;
```

程序执行后输出结果如图 5-11 所示。

输出 - （无标题）

SAS 系统　　2012年06月01日

Obs	x	x_sqrt	x_ceil	x_floor	x_int
1	4.0	2.00000	4	4	4
2	8.1	2.84605	9	8	8
3	8.2	2.86356	9	8	8
4	9.5	3.08221	10	9	9

图 5-11　对变量 x 进行函数处理

【程序解读】

① x_sqrt=sqrt(x)：对 x 求平方根。

② x_ceil=ceil(x)：对 x 求比 x 数值大的最小整数。

③ x_floor=floor(x)：对 x 求比输入值 x 小的最大整数。

④ x_int=int(x)：对 x 求整数。

6）LOG 函数、LOG2 函数和 LOG10 函数。

LOG 函数、LOG2 函数和 LOG10 函数这 3 个函数属于一类对数函数，只是底数不同。

LOG(x)：以 e 为底的自然对数。

LOG2(x)：以 2 为底的 x 的对数。

LOG10(x)：以 10 为底的 x 的对数。

【例 5.21】 对变量 x 取对数。

```
data  sy;
Input  id  x  times ;
   x_log=log(x);/*把 x 转化为以 e 为底的对数*/
   x_log2=log2(x);/*把 x 转化为以 2 为底的对数*/
   x_log10=log10(x);/*把 x 转化为以 10 为底的对数*/
cards;
101 2 1
102 4 2
;
run;
/*打印数据集到输出窗口，查看结果*/
proc  print  data=sy;
run;
```

程序执行后的输出结果如图 5-12 所示。

输出 - （无标题）

SAS 系统　　2012年06月01日

Obs	id	x	times	x_log	x_log2	x_log10
1	101	2	1	0.69315	1	0.30103
2	102	4	2	1.38629	2	0.60206

图 5-12　对变量 x 取对数

【程序解读】

① x_log=log(x)：把 x 转化为以 e 为底的对数。

② x_log2=log2(x)：把 x 转化为以 2 为底的对数。

③ x_log10=log10(x)：把 x 转化为以 10 为底的对数。

7）EXP 函数。

对于实际开发中以 e 为指数的幂级可以考虑用 EXP 函数转化数据，便于过程步调用统计分析过程对数据分析。

语法格式：EXP(argument)

【语法解读】 argument：为输入的数值参数。

功能：求以常值e为指数的 argument次 幂的值。

【例 5.22】 求以 e 为底的 x 次幂。

```
data  sy_exp;
input  x ;
  x_exp=exp(x);/*以 e 为底的 x 次幂*/
cards;
```

```
2
4
;
run;
```

【程序解读】

x_exp=exp(x)：求以 e 为底的 x 次幂。

（3）日期和时间函数

SAS 系统内部编译好了日期和时间处理函数，这些函数主要用来处理实际开发中的日期和时间值。常用日期和时间函数如表 5-3 所示。

表 5-3　常用日期和时间函数

日期和时间函数	功 能 说 明
YEAR	返回日期的年份
MONTH	返回日期的月份
DAY	返回日期的日
MDY	从年月日中返回日期值
WEEKDAY	从日期中求出星期几
QTR	根据日期求出季度
HMS	根据输入时间值返回一个 SAS 时间
DHMS	根据输入日期时间值返回一个 SAS 日期时间
DATEPART	根据输入日期时间值返回一个 SAS 日期部分
INTCK	返回两个日期按指定间隔求出的间隔值
TODAY	返回当前日期值
DATETIME	返回当前 SAS 日期时间值

为便于读者对每一个函数的应用理解，下面分别对每一个函数进行讲解最后再通过一个综合应用对这些函数进行联合应用的讲解。

1）YEAR 函数、MONTH 函数和 DAY 函数。

YEAR 函数、MONTH 函数和 DAY 函数这 3 个函数是对日期处理经常用到的函数，分别表示取出日期的年份、月份和日。

YEAR函数 语法格式：YEAR(date)

【语法解读】 date：为输入的日期值。

功能：返回 date日期 的年份。

MONTH函数语法格式：MONTH(date)

【语法解读】 date：为输入的日期值。

功能：返回 date日期 的月份。

DAY函数语法格式：DAY(date)

【语法解读】 date：为输入的日期值。

功能：返回 date日期 的日。

【例 5.23】 YEAR 函数、MONTH 函数和 DAY 函数 3 个函数联合应用拼接文件日期的实

际开发应用。

程序如下：

```
/*业务数据日期处理应用*/
%let  v_curr_dt=%sysfunc(date());  /*取当前日期*/
%let  v_year = %sysfunc(year(&v_curr_dt)); /*取当前年份*/
%let  v_month = %sysfunc(month(&v_curr_dt)); /*取当前月份*/
%let  v_day = %sysfunc(day(&v_curr_dt)); /*取当前日*/
%let  v_c_date = %eval(&v_year*10000 + &v_month*100 + &v_day);  /*实际数据业务日期*/
%put  &v_c_date;  /*实际数据业务日期输出到日志窗口*/
```

【程序解读】

① v_curr_dt=%sysfunc(date())：取当前系统日期赋值给宏变量。

② v_year = %sysfunc(year(&v_curr_dt))：取当前系统日期年份。

③ v_month = %sysfunc(month(&v_curr_dt))：取当前系统日期月份。

④ v_day = %sysfunc(day(&v_curr_dt))：取当前系统日期的日。

⑤ v_c_date = %eval(&v_year*10000 + &v_month*100 + &v_day)：根据实际业务日期拼接日期为 yyyymmdd 格式日期。

2）MDY 函数。

语法格式：MDY(month,day,year)

【语法解读】

month：为输入的月份，从 1～12 月的整数值。

day：为输入的天数，从 1～31 的整数值。

year：为输入的两位或 4 位的年份整数值。

功能：从 month 、day、 year 中返回一个 SAS 日期值。

【例 5.24】 批量读取 3 个连续日期下的文件，数据集命名为 lxfil_yyyymmdd 格式，存储到 d:\jx 文件夹。

```
libname  jx  'd:\jx';
%macro  readfile();
%let  path=D:\jx;
%let  beg_dt=%sysfunc(mdy(6,01,2012));/*定义开始日期*/
%let  end_dt=%sysfunc(mdy(6,03,2012));/*定义结束日期*/
%do  i=&beg_dt.  %to  &end_dt.;
    %let  v_dt=%sysfunc(compress(%sysfunc(putn(&i.,yymmdd10.)),-));
     %let  path= D:\jx\f_&v_dt;  /*定义外部文件路径*/
     %let  type=.txt;
filename  fil  "&path.&type.";
data  jx.lxfil_&v_dt.;/*生成 3 个带日期的数据集*/
    infile  fil  ;
    input  id
           name $;
run;
 %end;
```

```
%mend;
%readfile();   /*调用宏过程*/
```

【程序解读】

① %let beg_dt=%sysfunc(mdy(6,01,2012))：利用 MDY 函数取定义的开始日期。

② %let end_dt=%sysfunc(mdy(6,03,2012))：利用 MDY 函数取定义的结束日期。

③ %do i=&beg_dt. %to &end_dt.：循环控制语句。

④ %let v_dt=%sysfunc(compress(%sysfunc(putn(&i.,yymmdd10.)),-))：处理日期生成 yyyymmdd 格式。

⑤ data jx.lxfil_&v_dt.：数据步名为宏变量名的组合，生成 lxfil_yyymmdd 格式数据集。

3）WEEKDAY 函数。

WEEKDAY 函数返回一个日期的星期几。

语法格式：WEEKDAY（v_date）

【语法解读】 v_date 为输入日期。

功能：从输入日期输出星期几，1=Sunday, 2=Monday,…, 7=Saturday。

【例 5.25】 求当前日期星期几。

```
data   _null_;
   y=date();/*取当前日期*/
     v_weekday=weekday(y);/*返回星期几*/
     put   v_weekday;
run;
```

【程序解读】

v_weekday=weekday(y)：返回 y 日期星期几。

4）QTR 函数。

语法格式：QTR（v_date）

【语法解读】 v_date：为输入日期。

功能：根据输入日期返回一个数值，1 代表第一季度，2 代表第二季度，3 代表第三季度，4 代表第四季度。

在一些按季度结算的业务中经常用 QTR 函数来求出日期对应一年的第几个季度。

【例 5.26】 根据不同的季度调整价格，求出每季度应收入的钱数。

```
%let   path= D:\jx\price;   /*定义外部文件路径*/
%let   type=.txt;
%let   fil= "&path&type";
libname   jx   'd:\jx';   /*定义逻辑库*/
data   jx.price(drop=bh v_dt);   /*数据集存储到指定逻辑库*/
       Infile     &fil ;
       Informat     xs_dt    date8.;
       Input       bh              :$5.
             sc_name          :$8.
             xs_dt             :8.     /*销售日期*/
             amt               :6.
```

```
            ;
        v_dt= input(put(xs_dt, 10.), yymmdd10.);/*对数值日期转换为日期格式*/
        qtr=qtr(v_dt);/*根据日期求出对应的季度*/
    select  (qtr);
/*QTR 季度值求出对应价格*/
            When  (1)  price=6 ;
            When  (2)  price=5.2;
            When  (3)  price=4.8;
            otherwise   price=7.1;
end;
    sr_money=amt * price;  /*应收钱数*/
        format   v_dt  yymmdd10.;
run;
```

【程序解读】

① qtr=qtr(v_dt)：求出日期的季度。

② select 语句根据 QTR 季度变量赋值不同的 price（价格）。

5）HMS 函数和 DHMS 函数。

HMS 函数是对时间处理的函数，返回一个 SAS 时间。DHMS 函数只是多了一个日期参数，其他与 HMS 函数相同。DHMS 函数包含日期和时分秒。

HMS 函数语法格式：HMS（v_hour,v_minute,v_second）

【语法解读】

v_hour：为输入小时的数值。

v_minute：为输入分钟的数值。

v_second：为输入秒的数值。

功能：根据输入时间数值返回一个 SAS 时间。

DHMS 函数语法格式：DHMS（v_date,v_hour,v_minute,v_second）

【语法解读】

v_date：为输入日期的数值。

v_hour：为输入小时的数值。

v_minute：为输入分钟的数值。

v_second：为输入秒的数值。

功能：根据输入日期时间数值返回一个 SAS 日期时间。

【提示】 DHMS 函数与 HMS 函数属于同一类函数，可以对比学习其不同之处。

【例 5.27】 根据输入的时分秒取出时间格式。

```
data  datetime;
    input   v_date  v_hour  v_min  v_sec;
    v_time=hms(v_hour,v_min,v_sec);/*HMS 函数根据输入的时分秒取出时间*/
    v_dt= input(put(v_date, 10.), yymmdd10.);/*对数值日期转换为日期格式*/
    v_datetime=dhms(v_dt,v_hour,v_min,v_sec);/*DHMS 函数根据输入的日期和时分秒返回日期时间*/
    format  v_time  time.  v_datetime  datetime.;  /*定义输出格式*/
cards;
```

```
20100323 10 32 43
20100323 10 36 33
;
run;
```

【程序解读】

① v_time=hms(v_hour,v_min,v_sec)：通过 HMS 函数对输入的时分秒取出时间，赋值给变量 v_time。

② v_datetime=dhms(v_dt,v_hour,v_min,v_sec)：DHMS 函数根据输入的日期和时分秒返回日期时间。

6）DATEPART 函数。

对于实际业务需求如果只是需要日期部分而不需要时间部分可以应用 DATEPART 函数取对应日期。

语法格式：DATEPART（v_datetime）

【语法解读】 v_datetime：为输入的日期时间。

功能：根据输入日期时间返回一个 SAS 日期部分。

【例 5.28】 对例 5.27 进行改造。

```
data    datepart;
   input    v_date    v_hour    v_min    v_sec;
   v_dt= input(put(v_date, 10.), yymmdd10.);/*对数值日期转换为日期格式*/
   v_datetime=dhms(v_dt,v_hour,v_min,v_sec);/*DHMS 函数根据输入的日期和时分秒返回日期时间*/
   v_datepart=datepart(v_datetime);/*取日期部分*/
     format    v_datetime    datetime.   v_datepart    yymmdd10.; /*定义输出格式*/
cards;
20100323 10 32 43
20100323 10 36 33
;
run;
```

【程序解读】

v_datepart=datepart(v_datetime)：根据输入的日期时间变量取日期部分。

7）INTCK 函数。

对于金融业经常用 INTCK 函数求两个日期之间间隔的年数、月数。

语法格式：INTCK（date_interval,v_startdate,v_enddate）

【语法解读】

date_interval：为输入的日期间隔值，如 DAY、WEEK、MONTH、QTR、YEAR.、HOUR 等。

v_startdate：输入的开始日期。

v_enddate：输入的结束日期。

功能：返回两个日期按间隔求出的间隔值。

【例 5.29】 根据客户刷卡交易额，求出到账单还款日没有还款，预期的利息。

```
data card_inerest;
   input   id   name   $   tras_money   begin_dt   date7.   rate;
   days=intck('day',begin_dt,'31mar12'd);/*INTCK 函数求出预期的天数*/
   v_inerest=tras_money*rate*days; /*收取客户利息*/
cards;
10001  马小红  2000 08jan12 0.52
10002  杨小红  3500 08feb12 0.62
;
run;
```

【程序解读】

days=intck('day',begin_dt,'31mar12'd)：通过 INTCK 函数根据账单日和账单还款结束日期求出预期的天数。

8）TODAY 函数。

TODAY 函数返回系统当前日期。

语法格式：TODAY()

功能：返回当前 SAS 日期值。

【例 5.30】 TODAY 函数返回当前系统日期。

```
*today 函数返回当前日期;
data   today;
   v_today=today();   /*TODAY 函数返回当前日期*/
   format   v_today   yymmdd10. ;
run;
```

【程序解读】

① v_today=today()：TODAY 函数返回当前日期。

② format   v_today   yymmdd10. ：语句定义其输出日期格式。

9）DATETIME 函数。

DATETIME 函数返回当前 SAS 日期时间值。与 TODAY 函数对比可以发现，其不同点是 DATETIME 函数具有日期和时间值。

语法格式：DATETIME()

功能：返回当前 SAS 日期时间值。

【例 5.31】 DATETIME 函数应用，请取出当前 SAS 系统的日期时间。

程序如下：

```
*datetime 函数返回当前日期;
data   datetime;
   v_datetime=datetime();   /*DATETIME 函数返回当前日期时间*/
   format   v_datetime   datetime. ;
run;
```

【程序解读】

v_datetime=datetime()：DATETIME 函数返回系统当前日期时间。

（4）统计函数

对于统计专业而言，经常需要对实际数据做数理统计，SAS 系统也考虑到了这一点，内部编译好了很多统计函数，来满足统计学的实际应用。常用统计函数如表 5-4 所示。

表 5-4　常用统计函数

统计函数名	功 能 说 明
MEAN	计算几何平均数
PDF	计算分布密度
PROBNORM	计算标准正态分布概率
VAR	计算方差
STD	计算标准差
STDERR	计算均值估计的标准误差
CV	计算变异系数
SKEWNESS	计算偏斜度
KURTOSIS	计算峰度值
USS	计算未校正的平方和
CSS	计算修正的离差平方和
RANGE	计算最大值和最小值之间的极差

1）MEAN 函数。

语法格式：MEAN（v_num1,v_num2,⋯,v_numN）

【语法解读】

v_num1,v_num2..,v_numN：为输入的数值。

功能：对输入数值求几何平均数，为（v_num1+v_num2+⋯+v_numN）/N 的几何平均数。

【例 5.32】 MEAN 函数应用，求出 maths、English、Chinese 这 3 门课程的几何平均值。

程序如下：

```
data   score;
 Input  id  name  $  maths  English  Chinese;
   v_mean=mean(maths,english,chinese);   /*求几何平均数*/
cards;
1001  高宏  78 89 60
1002  马小名  88 67 79
1003  杨小和  98 87 86
;
run;
```

【程序解读】

v_mean=mean(maths,english,chinese)：对 3 科成绩求几何平均数。

2）PDF 函数和 PROBNORM 函数。

PDF 函数计算分布密度，PROBNORM 函数计算标准正态分布概率，两者都是计算分布。

PDF 语法格式：PDF(‘v_dist’,v_parm1,⋯,v_parmN)

【语法解读】

v_dist：确定分布状态，取值如'BERNOULLI'、'BETA'、'BINOMIAL'、'CAUCHY'、'CHISQUARE'、'EXPONENTIAL'、'F'、'GAMMA'、'GEOMETRIC'、'HYPERGEOMETRIC'、'LAPLACE'、'LOGISTIC'、'LOGNORMAL'、'NEGBINOMIAL'、'NORMAL'|'GAUSS'、'NORMALMIX'、'PARETO'、'POISSON'、'T'、'UNIFORM'、'WALD'|'IGAUSS'、'WEIBULL'。

v_parm1,⋯,v_parmN：为参数表，根据形状、位置或范围设置的参数。

功能：求分布密度值。

PROBNORM 函数语法格式：PROBNORM（v_y）

【语法解读】 v_y：为输入的随机变量数值。

功能：求标准正态分布的概率。

【例 5.33】 PDF 函数应用，求出分布状态为“BETA”、随机变量为 8、自由度为 2、分布密度为 1 的分布密度值；PROBNORM 函数应用，求出标准正态分布观测值小于或等于 1.6 的概率值。

程序如下：

```
data   pdf_probnorm;
        pdfchf=pdf('BETA',8,2,1);
/*PDF 函数求卡方分布，8 为随机变量，2 为自由度，1 为分布密度*/
        probnorm=probnorm(1.6);
      /*标准正态分布的观察值小于或等于 1.6 的概率值*/
run;
```

【程序解读】

① pdfchf=pdf('BETA',8,2,1)：PDF 函数求卡方分布，8 为随机变量，2 为自由度，1 分布密度。

② probnorm=probnorm(1.6)：标准正态分布的观察值小于或等于 1.6 的概率值。

3）VAR 函数、STD 函数和 STDERR 函数。

VAR 函数属于统计学中的计算方差函数，STD 函数属于统计学中的计算标准差函数，STDERR 函数为计算均值估计的标准误差函数。

VAR 函数语法格式：VAR(x1,x2,⋯,xn)

【语法解读】 x1,x2, ...,xn：为输入的数值。

功能：求方差。

STD 函数语法格式：STD(x1,x2,⋯,xn)

【语法解读】 x1,x2,⋯,xn：为输入的数值。

功能：求标准差。

STDERR 函数语法格式：STDERR(x1,x2,⋯,xn)

【语法解读】 x1,x2,⋯,xn：为输入的数值。

功能：求均值估计的标准误差。

【例 5.34】 VAR 函数求出 2、4 和 8 的方差，STD 函数求出 2、4 和 8 的标准差，STDERR 函数求出 2、4 和 8 的标准误差。

```
data   fun;
```

```
        varf=var(2,4,8);
        /*VAR 求 2、4 和 8 的方差*/
          stdf=std(2,4,8);
        /*STD 求 2、4 和 8 的标准差*/
        stderrf=stderr(2,4,8);
        /*STDERR 求 2、4 和 8 的均值估计的标准误差*/
    run;
```

【程序解读】

① varf=var(2,4,8)：VAR 统计函数求 2、4 和 8 的方差。

② stdf=std(2,4,8)：STD 函数求 2、4 和 8 的标准差。

③ stderrf=stderr(2,4,8)：STDERR 统计函数求 2、4 和 8 的均值估计的标准误差。

4）CV 函数。

CV 函数计算变异系数。

语法格式：CV(x1,x2,⋯,xn)

【语法解读】 x1,x2,⋯,xn：为输入的数值。

功能：求变异系数。

【例 5.35】 CV 函数求出 2、4 和 8 的变异系数。

```
    *CV 函数求变异系数;
    data  cv;
        cvf=cv(2,4,8);
        /*CVH 函数求 2、4 和 8 的变异系数*/
    run;
```

5）SKEWNESS 函数和 KURTOSIS 函数。

SKEWNESS 函数计算偏斜度，KURTOSIS 函数计算峰度值。

SKEWNESS 函数语法格式：SKEWNESS (x1,x2,⋯,xn)

【语法解读】 x1,x2,⋯,xn：为输入的数值。

功能：求偏斜度。

KURTOSIS 函数语法格式：KURTOSIS(X1,X2,⋯,XN)

【语法解读】 x1,x2,⋯,xN：为输入的数值。参数至少有 4 个值。

功能：求峰度值。

【例 5.36】 SKEWNESS 函数求出 2、4 和 8 的偏斜度，KURTOSIS 函数求出 2、4、5 和 8 的峰度值。

程序如下：

```
    data  ske_kur;
        skewf=skewness(2,4,8);
        /*SKEWNESS 求 2、4 和 8 的偏斜度*/
          kurtosisf=kurtosis(2,4,5,8);
        /*KURTOSIS 求 2、4、5 和 8 的峰度*/
    run;
```

【程序解读】

① skewf=skewness(2,4,8)：SKEWNESS 函数求 2、4 和 8 的偏斜度。

② kurtosisf=kurtosis(2,4,5,8)：KURTOSIS 求 2、4、5 和 8 的峰度。

6）USS 函数和 CSS 函数。

USS 函数计算未校正的平方和，CSS 函数计算修正的离差平方和。

USS 函数语法格式：USS(X1,X2,…,XN)

【语法解读】 x1,x2, ...,xN：为输入的数值。至少有一个非缺失值。

功能：求未校正的平方和。

CSS 函数语法格式：CSS(X1,X2,…,XN)

【语法解读】 x1,x2, ...,xn 为输入的数值。至少有一个非缺失值。

功能：求修正的离差平方和。

【例 5.37】 USS 函数应用，求出 2、4、6 和 8 未校正的平方和；CSS 函数应用，求出 2、4、6 和 8 修正的离差平方和。

程序如下：

```
data   uss_css;
       ussf=uss(2,4,6,8);
       /*USS 求 2、4、6 和 8 未校正的平方和*/
         cssf=css(2,4,6,8);
       /*CSS 求 2、4、6 和 8 修正的离差平方和*/
run;
```

【程序解读】

① ussf=uss(2,4,6,8)：USS 函数求 2、4、6 和 8 未校正的平方和。

② cssf=css(2,4,6,8)：CSS 函数求 2、4、6 和 8 修正的离差平方和。

7）RANGE 函数。

RANGE 函数计算最大值和最小值之间的极差。

语法格式：RANGE (X1,X2,…,XN)

【语法解读】 x1,x2,…,xN：为输入的数值。至少有一个非缺失值。

功能：求最大值和最小值之间的极差。

【例 5.38】 RANGE 函数应用，求出–2、4、6 和 9 一组数据中最大值和最小值之间的极差。

程序如下：

```
*range 函数求最大值和最小值之间的极差;
data   range;
        range=range(-2,4,6,9);
        /*RANGE 求–2、4、6 和 9 中最大值和最小值之间的极差*/
run;
```

【程序解读】

range=range(-2,4,6,9)：RANGE 统计函数求–2、4、6 和 9 中最大值和最小值之间的极差。

5.1.2 数据步引用函数

对于 SAS 系统而言，大部分的数据处理是在数据步，函数在数据步应用的概率比较高。根据业务需求通过数据步引用函数对变量进行处理，才能满足需求。

【例 5.39】 处理客户交易数据文件，并取出区号、电话号码、出生年份。

```
*文件数据字段根据业务需求处理字段，生成符合业务需求的数据集;
%let   path= D:\jx\jiaoyi;   /*定义外部文件路径*/
%let   type=.dat;
%let   filename= "&path&type";
libname   jx   'd:\jx';   /*定义逻辑库*/
data   jx.jiaoyi(keep=zone tel name ed jy_address birthday);   /*数据集存储到指定逻辑库*/
        Infile   &filename   dlm='|'   dsd   missover     ;
                input   phone      :$12.
                        name      :$8.
                        dq         :$4.
                        sf         :$18.
                        ed         :6.
                        jy_address :$20.
                  ;
                length zone $3.; /*定义了一个业务需求变量区号*/
                zone= scan(phone,1,'#');
/*根据业务需求用 SAS 内部函数 scan 处理字段 phone，取出区号*/
  length   tel   $8.; /*定义了一个业务需求变量电话号码*/
                tel= scan(phone,-1,'#');
/*根据业务需求用 SAS 内部函数 scan 处理字段 phone，取出电话号码*/
Length   birthday   $8.; /*定义了一个业务需求变量出生日期*/
      birthday= substr(sf,7,8);
/*根据业务需求用 SAS 内部函数 substr 截取字段身份证号，取出出生年份*/
run;
```

【程序解读】

1）zone= scan(phone,1,'#')：通过 scan 函数取出区号。

2）tel= scan(phone,-1,'#')：通过 scan 函数取出电话号码。

3）birthday= substr(sf,7,8)：通过 substr 函数取出出生年份。

5.1.3 宏过程引用函数

宏过程引用函数是通过宏过程对功能封装，其内部数据处理对函数应用，根据业务需求，对变量处理，从而满足业务需求。宏过程第 6 章会单独讲解。

【例 5.40】 对例 5.39 进行改造，应用宏过程实现。

```
%macro jiaoyi(v_filname,v_type);/*定义文件名变量和文件类型*/

*文件数据字段根据业务需求处理字段，生成符合业务需求的数据集;
%let   path= D:\jx\&v_filname;   /*定义外部文件路径*/
%let   type=&v_type;
```

```
%let    filename= "&path&type";
libname    jx 'd:\jx';    /*定义逻辑库*/
data    jx.jiaoyisj(keep=zone tel name ed jy_address birthday);    /*数据集存储到指定逻辑库*/
        Infile    &filename    dlm='|'    dsd    missover    ;
                input    phone      :$12.
                         name     :$8.
                         dq          :$4.
                         sf          :$18.
                         ed          :6.
                         jy_address :$20.
                     ;
                length zone $3.; /*定义了一个业务需求变量区号*/
                zone= scan(phone,1,'#');
/*根据业务需求用 SAS 内部函数 scan 处理字段 phone，取出区号*/
  length    tel    $8.; /*定义了一个业务需求变量电话号码*/
                tel= scan(phone,-1,'#');
/*根据业务需求用 SAS 内部函数 scan 处理字段 phone，取出电话号码*/
length    birthday    $8.; /*定义了一个业务需求变量出生日期*/
      birthday= substr(sf,7,8);
/*根据业务需求用 SAS 内部函数 substr 截取字段身份证号，取出出生年份*/
run;
%mend    jiaoyi;
%jiaoyi(jiaoyi,.dat);
```

【程序解读】

1）%macro jiaoyi(v_filname,v_type)：%macro 宏过程开始语句，jiaoyi 定义宏名，v_filname 和 v_type 为宏参数。

2）zone= scan(phone,1,'#')：通过 scan 函数取出区号。

3）tel= scan(phone,-1,'#')：通过 scan 函数取出电话号码。

4）birthday= substr(sf,7,8)：通过 substr 函数取出出生年份。

5）%mend jiaoyi:%mend：宏结束标志。

6）%jiaoyi(jiaoyi,.dat)：%宏名为宏调用，传递实参 jiaoyi 和.dat 给所定义的宏变量。

5.1.4　函数综合应用

实际的业界开发中在宏变量、数据步和宏过程中都会应用到函数，综合应用是实际业界开发中经常用到的。

【例 5.41】 根据数据文件日期处理相应数据，批量生成对应日期的数据，存储到 d:\jx 目录。批量读取 d:\jx 目录下的 3 个文件 jiaoyi_20120601.dat、jiaoyi_20120602.dat 和 jiaoyi_20120603.dat。

```
/*批量读取 d:\jx 目录下的 3 个文件 jiaoyi_20120601.dat、jiaoyi_20120602.dat 和 jiaoyi_20120603.dat*/
libname    jx 'd:\jx';
/*批量处理按日期命名的文件*/
%macro    plfile(v_type);
%let    path=D:\jx;
```

```
%let    beg_dt=%sysfunc(mdy(6,01,2012));/*定义开始日期*/
%let    end_dt=%sysfunc(mdy(6,03,2012));/*定义结束日期*/
%do    i=&beg_dt.    %to    &end_dt.;
     %let    v_dt=%sysfunc(compress(%sysfunc(putn(&i.,yymmdd10.)),-));
       %let    path= D:\jx\jiaoyi_&v_dt;    /*定义外部文件路径*/
       %let    type=&v_type;
filename    fil    "&path.&type.";
data    jx.jiaoyisj_&v_dt(keep=zone tel name ed jy_address birthday);    /*数据集存储到指定逻辑库*/
         Infile    fil    dlm='|'    dsd    missover      ;
                  input    phone       :$12.
                              name      :$8.
                              dq           :$4.
                              sf           :$18.
                              ed           :6.
                              jy_address :$20.
                        ;
                  length zone $3.; /*定义了一个业务需求变量区号*/
                  zone= scan(phone,1,'#');
/*根据业务需求用 SAS 内部函数 scan 处理字段 phone，取出区号*/
     length    tel    $8.; /*定义了一个业务需求变量电话号码*/
                  tel= scan(phone,-1,'#');
/*根据业务需求用 SAS 内部函数 scan 处理字段 phone，取出电话号码*/
length    birthday    $8.; /*定义了一个业务需求变量出生日期*/
       birthday= substr(sf,7,8);
/*根据业务需求用 SAS 内部函数 substr 截取字段身份证号，取出出生年份*/
run;
  %end;
%mend;
%plfile(.dat);/*调用宏过程*/
```

【程序解读】

1）%do i=&beg_dt. %to &end_dt.：根据定义的日期变量循环执行。

2）jx.jiaoyisj_&v_dt：生成的数据集为 jiaoyisj_yyyymmdd 格式数据集名。

3）zone= scan(phone,1,'#')：通过 scan 函数取出区号。

4）tel= scan(phone,-1,'#')：通过 scan 函数取出电话号码。

5）birthday= substr(sf,7,8)：通过 substr 函数取出出生年份。

6）%plfile(.dat)：调用宏过程，传递实参.dat。

5.2 信用卡收入分析案例

中国信用卡市场目前处于高速发展期。信用卡收入的主要来源有刷卡佣金、年费、分期付款费、取现等。

1）发卡行收取开户用户的发卡费，如每张 100 元，包括挂失、损失重新补办卡等。

2）透支利息收入，透支利息收入利率很高，每天万分之五至千分之一。

3）向特约商户收取的佣金是信用卡对特约商户的费用收取。

4）信用卡取现，如跨行取现、跨地区取现的费用。

5）向客户递送对账单，对账单附带的广告收入。

6）商家酒店、航空公司向持卡客户推出打折服务，之后向商家收取的介绍费。

【例 5.42】 某客户账单日为 2011-12-07，到期还款日为 2011-12-26，上期应还款额为 1881.32，上期已还款额 1900.00，本期新增金额 1728.56，求本期应还款额和最低还款额。

【分析】

1．客户应还款额

每个客户的信用卡都有一个账单日，如本案例客户账单日为每月的 8 号，那么每月的 7 号是其出账单的对账日，这一天客户应还款额通过银行内部程序计算出来。

客户应还款额公式如下：

客户应还款额=上期应还款额–上期已还款额+本期新增金额=本期应还款额

2．最低还款额

信息卡客户如果在到期还款日前全额偿还所欠银行款项有困难，可按照发卡机构规定的最低还款额还款。

最低还款额的计算方法如下：

最低还款额=以前最低还款额累计未还部分+本月取现及转账贷款未还部分+本月超限额消费贷款+所有未还的限额内消费贷款×10%。

按最低还款额规定还款的，发卡机构只对未清偿部分每日按万分之五计收从银行记账日起至还款日止的贷款利息，贷款利息按月计收复利。

最低还款额即使用循环信用时需要偿还的最低金额，不低于欠款余额的 10%。以对账单通知金额为准。

如果在到期还款日之前归还金额大于或等于最低还款额，但低于本期应还金额时，只需支付利息；如果低于最低还款额，则除了利息外，还要按最低还款额未还部分的 5%支付滞纳金，并会对客户的信用记录造成影响。

如果客户在到期还款日前全额还款，则不用支付利息，享受免息还款期优惠。免息还款期优惠只针对消费交易，从交易日起至到期还款日之间的日期为免息还款期，最长 56 天，最短 25 天。系统按年自动扣收的年费也享受免息还款期。

```
*信用卡客户账单日还款计算宏过程;
%macro  Repayment(v_previous,v_payment,v_newactivity,v_minpre,v_monthcurr,v_bycx,v_whxe);
data  custer;
   v_currentb=sum(&v_previous,-&v_payment,&v_newactivity); /*本期应还款额*/
   v_minpayment=sum(&v_minpre,&v_monthcurr,&v_bycx,&v_whxe*0.1);/*最低还款额*/
 run;
 Proc  print  data=custer  lable;
 lable  v_currentb='本期应还款额'  v_minpayment='最低还款额';
 run;
%mend  Repayment;
%Repayment(1881.32,1900,1728.56,0,0,0,1728.56);/*调用宏过程*/
```

【程序解读】

1）v_currentb=sum(&v_previous,-&v_payment,&v_newactivity);：v_currentb 为本期应还款额，v_previous 为上期应还款额，v_payment 为上期已还款额，v_newactivity 为本期新增金额。

2）v_minpayment=sum(&v_minpre,&v_monthcurr,&v_bycx,&v_whxe*0.1);：v_minpayment 为最低还款额，v_minpre 为以前最低还款额累计未还部分，v_monthcurr 为本月取现及转账贷款未还部分，v_bycx 为本月超限额消费贷款，v_whxe 为所有未还的限额内消费贷款。

程序执行后如图 5-13 所示。

输出 – （无标题）

SAS 系统　　2012年08

Obs	本期应还款额	最低还款额
1	1709.88	172.856

图 5-13　客户账单还款信息

第 6 章　宏基础与案例

6.1　宏基础

SAS 系统将一个变量、一段程序或者一个文本命名，供以后调用，称为宏。宏分为宏变量和宏过程，其中宏过程就是对一些 SAS 语句实现某些功能的封装。为便于某些功能的循环利用，通过宏过程可以调用封装在宏中的 SAS 语句。

6.1.1　宏概述与定义

1．宏过程概述

SAS 宏语言分为宏变量和宏过程。宏变量相当于其他语言定义的变量。宏过程是对 SAS 程序功能的封装，实现程序的重复利用，相当于关系数据库中的存储过程。每一个宏过程实现满足业务需求的不同功能，用户用到时只需要知道宏过程名即可。

宏的主要功能如下：

1）实现功能封装，用户用时直接调用宏过程名。

2）保持 SAS 程序的独立性和移植性，一段程序在多种情况下均可运行。

3）重复执行 SAS 程序。

4）获取 SAS 系统信息，SAS 启动时就创建了一些自动宏变量，用以存储当前 SAS 进程启动的日期、时间、版本号及其他信息，用户可以在任何情况下使用这些宏变量。

5）开发交互式系统。使用 SAS 宏语言的%Window 语句及一些基本的编程语句开发交互式用户界面。

6）宏功能可实现 SAS 程序的独立性和移植性，一段程序在多种情况下均可运行，得到期望的结果。

7）不同的 SAS 数据步或过程步之间传递数据，SAS 宏变量可在 SAS 的任何地方被引用，具有全局性，成为不同过程间传递数据的方法。

2．宏定义

宏定义分为宏变量定义和宏过程定义。

（1）宏变量定义

宏变量和数据步中的变量不同，宏变量可以在 SAS 程序的任何地方引用和定义，属于弱类型语句。数据步中定义的变量是与创建的数据集结合起来的，变量值与实际数据步中传递的观测有关。宏变量有两种：一种是用户定义的宏变量，另一种是 SAS 系统内部具有的自动宏变量。用户定义的宏变量是指用户根据需求定义的变量，对于用户定义的宏变量用户可以给变量赋值，也可以为空，可以指定范围，通过%GLOBAL 声名可以定义全局宏变量，默认是全局宏变量。

宏变量定义语法：%LET 宏变量名<= 赋实际值>;

【语法解读】

%LET：定义宏变量的关键字。

宏变量名：用户定义在宏语句中的变量名，命名遵守 SAS 命名规范。

<=赋实际值>：宏变量根据需求可以直接赋实际值，可选项。

【例 6.1】 %LET 语句定义宏变量 v_name。

```
%let   v_name;
```

【程序解读】%let 定义了宏变量 v_name。该宏变量虽然简单，但执行的宏机制流程不变。

对宏变量，如果需要赋实际值，可以直接赋值，不需要另定义，SAS 程序属于弱类型语言。

【例 6.2】 定义宏变量 v_total，并赋值“60”。

程序如下：

```
%let   v_str=60;
```

【程序解读】 定义宏变量 v_str，并给宏变量赋实际值，告诉宏解析器，此宏变量初值为“60”。

对于宏变量的引用，需要“&”符号，具体语法如下：

宏变量引用语法：&宏名

【提示】 间接引用宏变量要用两个&&符号，如&&v_var1.&v_var2，中间的点.属于间接引用分隔，告诉SAS系统是两个宏变量。&&v_var1 中的第一个“&”符号告诉SAS系统启动宏TOKEN解析器，然后读到&v_var1 引用此宏变量。

对于 SAS 系统内部具有的自动宏变量，可以通过这些变量查询一些系统信息，通过宏语句%PUT 可以直接调用这些自动宏变量。

【例 6.3】 调用自动宏变量，查询当前日期。

程序如下：

```
%put    &syslast;
```

【程序解读】 %PUT 语句可以直接调用自动宏变量，通过&符号加宏变量名的方式直接引用。

【提示】

1）对于自动宏变量_AUTOMATIC_引用时比较特殊，不需要符号&，直接引用，方式为%PUT _AUTOMATIC_。

2）对于文本串中引用宏变量，必须用双引号括起来，而不能用单引号。例如下面的程序：

```
%let    name=chiran;
DATA   fz;
str="My name is   &name"; /*引用宏变量 name*/
RUN;
```

3）宏变量中可以直接引用宏函数传递的值。语法如下：

```
%let v_var=%宏函数( );
```

（2）宏过程

宏过程是宏语言的核心，宏过程实现了程序的重复利用。对于宏过程而言用户所调用的宏过程都是已经编译好的，宏过程中包含一些程序的声名，接收外部传递的参数。

宏过程语法格式：%MACRO 宏名 <（宏参数）> </参数选项>;

宏功能语句;

%MEND <宏名>;

【语法解读】

%MACRO：定义宏过程语句开始的关键字，必选项，固定关键字。

宏名：给宏起的名字，命名要遵守命名规范。

宏参数：可选项，给宏定义的参数，多个参数用英文状态输入法下的逗号隔开。

参数选项：可选项，参数可以取 CMD、DES="text"、PARMBUFF、PBUFF、STMT、SOURCE、SRC、STORE。

宏功能语句：SAS 宏语句，实现某些功能的 SAS 语句，如数据步语句、过程步语句和宏变量定义语句。

%MEND：宏结束标志，必选项，固定关键字。

宏名：%MEND 语句中的宏名为宏开始处定义的宏名，此处告诉 SAS 系统宏过程定义结束。可选项，宏名此处可以省略。

调用宏过程方法：

```
%宏名<实参>;
```

【例 6.4】 编写一个宏过程，无参数宏过程，根据条件生成刷卡交易数据集。

```
%let   path= D:\jx\shuakajy;   /*定义外部文件路径*/
%let   type=.txt;
%let   fil= "&path&type";
libname   jx   'd:\jx';   /*定义逻辑库*/
%macro   shuaka();
data   jx.shuak3000(keep=name   month1   month2   month3   month4   tran_max   tran_min)
       jx.shuak5000(keep=name   month1   month2   month3   month4   tran_max   tran_min)
       jx.shuakother(keep=name month1 month2 month3 month4   tran_max   tran_min);
/*数据集存储到指定逻辑库*/
  infile   &fil ;
  input   id          :$9.
          name        :$8.
          month1      :8.
          month2      :8.
          month3      :8.
          month4      :8.
          ;
   length     tran_max    8.;
   tran_max=max(month1,month2,month3,month4);
/*MAX 函数求出客户 4 个月中刷卡交易的最高额度*/
 length     tran_min    8.;
```

```
    tran_min=min(month1,month2,month3,month4);
   /*MIN 函数求出客户 4 个月中刷卡交易的最低额度*/
if  month1>3000  and  month1<5000  then  output  jx.shuak3000;
       else  if  month1>5000  then  output  jx.shuak5000;
          else  output  jx.shuakother;
run;
%mend;
%shuaka();  /*调用宏过程*/
```

【程序解读】

1）宏过程 shuaka()没有宏参数，SAS 系统对编辑好的宏过程提交后先输入到缓冲区，然后通过字符扫描组件判断出是宏过程，调入 TOKEN 解析器，通过 TOKEN 解析器的 4 个组件解析宏语句，解析完成，传送到编译器编译程序，检查没有语法错误后，编译通过。

2）宏过程对数据步封装，数据步内部根据 if 语句处理数据。

3）if month1>3000 and month1<5000 then output jx.shuak3000：month1>3000 且 month1<5000 的数据生成数据集存储到数据集 jx.shuak3000。

4）else if month1>5000 then output jx.shuak5000：month1>5000 0 的数据生成数据集存储到数据集 jx.shuak5000。

5）else output jx.shuakother：不满足上述两个条件的数据集生成到数据集 jx.shuakother。

【例 6.5】 对例 6.4 进行改造，编写一个宏过程，有参数宏过程，根据条件生成刷卡交易数据集。

```
%let  path= D:\jx\shuakajy;  /*定义外部文件路径*/
%let  type=.txt;
%let  fil= "&path&type";
libname  jx 'd:\jx';  /*定义逻辑库*/
%macro  shuakaycs(v_min,v_max);/*定义宏形参 v_min 和 v_max*/
data  jx.shuaky3000(keep=name  month1 month2 month3 month4  tran_max  tran_min)
       jx.shuaky5000(keep=name  month1  month2  month3  month4  tran_max  tran_min)
       jx.shuakyother(keep=name month1 month2 month3 month4  tran_max  tran_min);
 /*数据集存储到指定逻辑库*/
      infile  &fil ;
        input  id          :$9.
              name         :$8.
              month1       :8.
              month2       :8.
              month3       :8.
              month4       :8.
                 ;
length  tran_max  8.;
    tran_max=max(month1,month2,month3,month4);
 /*MAX 函数求出客户 4 个月中刷卡交易的最高额度*/
length  tran_min  8.;
    tran_min=min(month1,month2,month3,month4);
   /*MIN 函数求出客户 4 个月中刷卡交易的最低额度*/
```

```
if   month1>&v_min   and   month1<&v_max   then   output   jx.shuaky3000;
     else   if   month1>&v_max   then   output   jx.shuaky5000;
       else   output   jx.shuakyother;
run;
%mend;
%shuakaycs(3000,5000);
/*调用宏过程，并传递实参 3000 和 5000 分别给形参 v_min 和 v_max*/
```

【程序解读】

1）%macro shuakaycs(v_min,v_max)：定义宏形参 v_min 和 v_max。

2）if month1>&v_min and month1<&v_max then output jx.shuaky3000：形参 v_min 和 v_max 传递给条件语句，month1>&v_min and month1<&v_max 时输出到数据集 jx.shuaky3000。

3）else if month1>&v_max then output jx.shuaky5000：month1>&v_max 时输出到数据集 jx.shuaky5000。

4）else output jx.shuakyother：上面条件不满足时其他情况输出到数据集 jx.shuakyother。

5）%shuakaycs(3000,5000)：调用宏过程，并传递实参 3000 和 5000 分别给形参 v_min 和 v_max。

【例 6.6】 对例 6.5 进行改造，编写一个宏过程，有参数宏过程和宏选项，根据条件生成刷卡交易数据集。

```
options   mstored   sasmstore=jx;
/*选项指定宏存储的逻辑库，宏过程中 stroe 选项使用时必须先指定此语句才能用*/
%let   path= D:\jx\shuakajy;   /*定义外部文件路径*/
%let   type=.txt;
%let   fil= "&path&type";
libname   jx   'd:\jx';   /*定义逻辑库*/
%macro   shuakayxx(v_min,v_max)/store secure;/*创建一个加密宏过程*/
data   jx.shuaky3000(keep=name   month1   month2   month3   month4   tran_max   tran_min)
       jx.shuaky5000(keep=name   month1   month2   month3   month4   tran_max   tran_min)
       jx.shuakyother(keep=name   month1   month2   month3   month4   tran_max tran_min) ;
 /*数据集存储到指定逻辑库*/
       infile   &fil ;
       input   id          :$9.
               name        :$8.
               month1       :8.
               month2       :8.
               month3       :8.
               month4       :8.
                ;
   length   tran_max   8.;
    tran_max=max(month1,month2,month3,month4);
/*MAX 函数求出客户 4 个月中刷卡交易的最高额度*/
   length   tran_min   8.;
    tran_min=min(month1,month2,month3,month4);
   /*MIN 函数求出客户 4 个月中刷卡交易的最低额度*/
```

```
if  month1>&v_min  and  month1<&v_max  then  output  jx.shuaky3000;
    else  if  month1>&v_max  then  output  jx.shuaky5000;
     else  output  jx.shuakyother;
run;
%mend;
%shuakayxx(3000,5000);
/*调用宏过程，并传递实参 3000 和 5000 分别给形参 v_min 和 v_max*/
```

【程序解读】

1）options mstored sasmstore=jx：选项指定宏存储的逻辑库，宏过程中 stroe 选项使用时必须先指定此语句才能用。

2）%macro shuakayxx(v_min,v_max)/store secure：创建一个加密宏过程，指定选项/store secure。

【提示】/store secure 选项使用时必须先指定 options mstored sasmstore=jx 语句，告诉 SAS 系统存储到哪个逻辑库，否则会报错，语句无效。

【例 6.7】 宏过程实现数据集 copy，根据传递参数表复制数据集到指定逻辑库。

```
libname  scr  'd:\jx';  /*原数据集逻辑库*/
libname  tar  'd:\jx\test';  /*目标数据集逻辑库*/
%macro cb(v_table_name);
proc  copy  in=scr  out=tar  noclone;  /*noclone 选项应用，不复制原数据集属性*/
  select  &v_table_name;
run;
%mend  cb;
%cb(custer);/*调用宏过程，传递实参数据集名 custer 给形参 v_table_name */
```

【程序解读】

1）宏过程把 copy 过程封装，用户只需要知道宏过程名 cb，调用此宏过程就可以。

2）%cb(custer)：调用编译好的宏过程，传递实参数据集名 custer 给形参 v_table_name。

6.1.2　宏过程应用

实际开发中宏过程应用的概率比较高。通过宏过程应用把数据步、过程步以及其他能用宏封装的 SAS 程序封装成一个过程。用户用到时直接调用，从而有利于程序的管理和安全，对于不懂 SAS 编程语言的用户，只需要告诉他这个宏过程名就可以直接应用，实现满足业务需求的功能。

【例 6.8】 编写一个实现加减乘除的宏过程。

```
*实现加减乘除运算的宏过程;
%macro  computer(v_jg,x,y);
   %if  &v_jg=add  %then %put %eval(&x+&y);
/*加法运算,%eval 宏函数计算非浮点型数据，实现加法运算*/
        %else  %do;
            %if  &v_jg=sub  %then  %put  %eval(&y-&x);
   /*求减法运算，%eval 宏函数计算非浮点型数据，实现减法运算*/
```

```
              %else    %do;
                      %if    &v_jg=div    %then    %put    %eval(&y/&x);
/*求除法运算，%eval 宏函数计算非浮点型数据，实现除法运算*/
                             %else    %put    %eval(&y*&x);
/*当上面条件都不满足时，实现乘法运算*/
                   %end;
         %end;
%mend    computer;
%computer(add,1,2);    /*调用宏实现加法运算*/
%computer(sub,1,7);    /*调用宏实现减法运算*/
%computer(div,2,8);    /*调用宏实现除法运算*/
%computer(mult,2,3);   /*调用宏实现乘法运算，这里参数可以除上面 3 种情况外任意写*/
```

【程序解读】

1）宏语句%if 语句为条件选择语句，宏语句以%开始。&v_jg=add 条件成立执行%then 语句，实现%eval(&x+&y)的加法运算。

2）%else %do %if &v_jg=sub %then %put %eval(&y-&x)：语句实现减法运算。

3）%else %do;%if &v_jg=div %then %put %eval(&y/&x)：语句实现除法运算。

4）%else %put %eval(&y*&x)：宏语句实现乘法运算。

【例 6.9】 宏变量为宏程序，将一段程序赋给一个宏变量 dayin。

```
* %STR 宏函数,将一段程序赋给一个宏变量 dayin；;
%let    dayin=%str(
proc    print    data=sashelp.adomsg;
run;
);
&dayin; /*调用宏变量*/
```

【程序解读】

1）通过%STR( )宏函数将一段程序赋给一个宏变量 dayin。

2）PROC print data=sashelp.adomsg;run：打印宏过程封装在宏函数%str()中，通过此过程封装。

3）&dayin：调用宏变量，实现打印功能。

【例 6.10】 宏过程实现条件过滤数据集，满足程序重复利用。取出数学成绩在 80～90，班级为 1 班的学生信息。

```
%macro    between(v_start,v_end,v_class);
/*宏形参 v_star 和 v_end 表示一个变量范围的开始和结束*/
%let    path= D:\jx\stuscore;    /*定义外部文件路径*/
%let    type=.txt;
%let    fil= "&path&type";
libname    jx    'd:\jx';    /*定义逻辑库*/
data    jx.stu_score;    /*数据集存储到指定逻辑库*/
            Infile      &fil    dlm='|'    dsd    missover    ;
            input           id                       :$4.
```

```
                    cust_name        :$8.
                    math              :3.
                    english           :3.
                    chinese           :3.
                    class             :1.
                ;
  length   stu_sum   5.;
  stu_sum=sum(math,english,chinese);/*求每位学生的总成绩*/
run;
data   stu&v_start;
 set   jx.stu_score (where=((math   between   &v_start   and   &v_end) and   class=&v_class));
 /*数学成绩在 v_start 和 v_end 范围之间的学生*/
run;
proc   print   data=stu&v_start;
run;
%mend   between;
%between(80,90,1);
```

【程序解读】

1）%macro between(v_start,v_end,v_class)：定义宏名 between，宏形参 v_star 和 v_end 表示一个变量范围的开始和结束，v_class 为班级。

2）set jx.stu_score (where=((math between &v_start and &v_end) and class=&v_class))：宏变量形参传递给 where 条件语句，math 成绩在 v_start 和 v_end 范围之间[v_start,v_end]，且班级 class 为 v_class 变量的值。

3）%between(80,90,1)：宏过程调用，传递 80 给变量 v_start，传递 90 给变量 v_end，传递 1 给变量 v_class。

【例 6.11】 宏过程中利用 firstobs= 和 obs=参数控制读取文件的条数，从数据文件的第 2 条记录读取到第 4 条记录结束。

```
 %macro   readfile(v_path,v_type);/*宏形参 v_path 和 v_type 分别表示文件路径和文件类型  */
%let   path=&v_path.;   /*定义外部文件路径*/
%let   type=&v_type;
%let   fil= "&path&type";
libname   jx   'd:\jx';   /*定义逻辑库*/
data   jx.stu_sj;   /*数据集存储到指定逻辑库*/
          Infile   &fil   dlm='|'    firstobs=2 obs=4   dsd   missover   ;
           input          id                      :$4.
                    cust_name               :$8.
                    math                    :3.
                    english                 :3.
                    chinese                 :3.
                    class                   :1.
                ;
run;
%mend   readfile;
```

```
%readfile(D:\jx\stuscore,.txt);
/*调用宏过程，传递实参路径和文件类型给变量 v_path 和 v_type*/
```

【程序解读】

宏过程封装了数据步处理数据的过程，“Infile &fil dlm='|' firstobs=2 obs=4 dsd missover ;”语句读取外部数据文件，firstobs=2 obs=4 选项分别控制读取文件从数据文件的第 2 条记录读取到第 4 条记录结束。

【例 6.12】 宏变量引用。

```
libname   jx   'd:\jx';   /*定义逻辑库*/
%let   v_name=stu;/*数据集变量名*/
%let   v_lib=jx;/*逻辑库变量名*/
%let   desc="the &v_name datasets";/*字符串中引用宏变量要放于双引号中*/
data     &v_lib..&v_name.;
/*进行层级引用，如库名.表名，库名为宏变量时，一定要在后面加两个“.”*/
  input   id   name   $;
cards;
1001  高小红
1002  杨小华
;
run;
```

【程序解读】

1）%let desc="the &v_name datasets"：宏变量在字符串中引用要放于双引号中。

2）DATA &v_lib..&v_name.：宏变量进行层级引用，如库名.表名，库名为宏变量时，一定要在后面加两个“.”。

【例 6.13】 数据集拆分，根据需求把数据集拆分成 3 个，第一个数据集从第 1 条记录至第 5 条记录，第二个数据集从第 6 条记录至第 10 条记录，第三个数据集从第 11 条记录到数据集结束。

```
/*拆分一个数据集*/
%macro   splitdata(v_first,v_obs);
  %if   &v_first<&v_obs   %then   %do;
    data   sp&v_first;
    set   sashelp.Bweight   (firstobs=&v_first   obs=&v_obs);
   run;
     %end;
     %else   %do;
     data   sp&v_first;
          set   sashelp.Bweight   (firstobs=&v_first);
   run;
    %end;
%mend;
%splitdata(1,5);
%splitdata(6,10);
%splitdata(11);/*只有 v_first 有值，取后面的全部*/
```

【程序解读】

1）宏过程实现数据集拆分，根据%if &v_first<&v_obs 条件语句判断，条件成立执行%do….%end 语句块中的创建数据集，条件不成立执行%else 语句中%do….%end 语句块中的创建数据集。

2）set sashelp.Bweight(firstobs=&v_first obs=&v_obs)：表示从数据集 firstobs=&v_first 条件的第&v_first 条记录取数，obs=&v_obs 表示取到变量传递的参数第&v_obs 条结束。

3）set sashelp.Bweight(firstobs=&v_first)：表示从数据集 firstobs=&v_first 条件的第&v_first 条记录取数，直到结束。

【例 6.14】 宏过程条件语句应用，根据不同的条件执行不同的宏过程。

```
    /* 先对外部数据文件处理*/
%macro sj(v_path,v_type);/*v_path:文件路径，v_type:文件类型*/
%let   path=&v_path.;   /*定义外部文件路径*/
%let   type=&v_type;
%let   fil= "&path&type";
libname   jx   'd:\jx';   /*定义逻辑库*/
data   jx.shuaka;   /*刷卡交易一月份数据*/
          Infile   &fil dlm='|'      dsd   missover   ;
          input                      id            :$4.
                                     amt           :6.
                                     year          :$8.
                ;
     length   month $2.;
      month=substr(year,5,2);
run;
%mend sj;
%sj(D:\jx\shuaka,.dat);/*调用宏过程，传递实参路径和文件类型给变量 v_path 和 v_type*/
/*刷卡交易分析宏*/
%macro   analy01;
proc   means   data=jx.shuaka(where=(month='01'));
 run;
%mend;
%macro   analy02;
proc   means   data=jx.shuaka(where=(month='02'));
run;
%mend;
%macro analyother;
proc   means   data=jx.shuaka(where=(month^='01'   and   month^='02' ));
run;
%mend;
%macro   analy(v_month);
%if   &v_month=01   %then   %analy01;
    %else   %if   &v_month=02   %then   %analy02;
       %else   %analyother;
%mend;
```

```
%analy(01);/*分析一月份刷卡交易*/
%analy(02);/*分析二月份刷卡交易*/
%analy(03);/*分析其他月份刷卡交易*/
```

【程序解读】

1）%macro analy01：分析一月份数据宏过程定义。

2）PROC means data=jx.shuaka(where=(month='01'))：根据 where 条件取一月份数据分析。

3）%macro analy02；分析二月份数据宏过程定义。

4）PROC means data=jx.shuaka(where=(month='02'))：根据 where 条件取二月份数据分析。

5）%macro analyother：其他月份数据分析。

6）%macro analy(v_month);定义一个主宏，此宏过程根据条件调其他宏过程，实现了宏过程嵌套利用。

7）%if &v_month=01 %then %analy01：&v_month=01 条件成立执行调用宏过程%analy01。

8）%else %if &v_month=02 %then %analy02：&v_month=02 条件成立调用宏过程%analy02。

9）%else %analyother：上述条件都不成立，执行调用宏过程%analyother。

【例 6.15】 根据宏过程中的条件选择不同的 SAS 内部过程，实现数据集打印和报表生成功能。

```
/*  先对外部数据文件处理*/
%macro    sj(v_path,v_type);/*v_path:文件路径，v_type:文件类型*/
%let    path=&v_path.;    /*定义外部文件路径*/
%let    type=&v_type;
%let    fil= "&path&type";
libname    jx    'd:\jx';    /*定义逻辑库*/
data    jx.shuaka;    /*刷卡交易一月份数据*/
            Infile    &fil    dlm='|'        dsd    missover    ;
             input                    id             :$4.
                                      amt            :6.
                                      year           :$8.
                  ;
        length    month $2.;
         month=substr(year,5,2);
run;
%mend    sj;
%sj(D:\jx\shuaka,.dat);/*调用宏过程，传递实参路径和文件类型给变量 v_path 和 v_type*/
/*根据条件选择打印功能或报表生成功能*/
%macro    choice(v_pro,v_data);
%if    &v_pro=print    %then    %do;
proc    print    data=&v_data;
run;
  %end;
%else    %if    &v_pro=report    %then    %do;
  options    date    number    ps=18    ls=80    fmtsearch=(jx);
 proc    report    data=&v_data nowd;
     Column    id    amt    year;
```

```
        Where   month='02';/*取 2 月份数据*/
          title 'credit cards reports';
run;
     %end;
%mend choice;
%choice(print,jx.shuaka);/*传递实参 print，调打印过程*/
%choice(report,jx.shuaka);/*传递实参 report，调生成报表过程*/
```

【程序解读】

1）%if &v_pro=print %then %do：传递参数为 print，符合条件执行打印功能。

2）%else %if &v_pro=report %then %do：传递参数为 report，符合条件执行报表生成功能。

【例 6.16】 判断指定的逻辑库下数据集是否存在。

```
*判断逻辑库下数据集是否存在;
libname   jx   'd:\jx';
%macro   pdsjj(v_dataset);
data   _null_;
  v_name="&v_dataset";
    if   (exist(v_name))   then   put   v_name '数据集已经存在';
          else   put   v_name '数据集不存在';
run;
%mend   pdsjj;
%pdsjj(jx.shuaka); /*宏调用，传递实参 jx.shuaka 给宏变量 v_dataset*/
```

【程序解读】

1）data _null_;：数据步只是做数据处理，不生成数据集，数据集名用_null_。

2）if (exist(v_name)) then put v_name '数据集已经存在';：此语句通过 exist 函数判断数据集是否存在。如果条件成立执行此语句，条件不成立执行“else put v_name '数据集不存在';语句”。

6.2 文件夹判断案例

实际开发应用中，数据文件或数据集有时需要放在 Windows 环境或 UNIX 环境的目录中。如果目录不存在，需要程序自动创建，实现了程序自动判断并创建目录的自动化程序，减少了人为的干预。

【例 6.17】 Windows 环境判断路径文件夹是否存在，不存在程序自动创建。

```
*路径文件夹创建与判断;
%let   commod=mkdir   d:\jx\kaifa;
%let   path=d:\jx\kaifa;
%macro   panduan;
%if   %sysfunc(fileexist(&path))   %then   %put '路径存在';
%else
%put   '路径文件夹不存在，需要执行下面语句创建';
X   &commod;   /*调用宏变量，创建路径文件夹*/
```

```
%mend;
%panduan;/*宏调用*/
```

程序执行后日志窗口显示如图 6-1 所示。

查看目录，显示路径文件夹已经自动创建，如图 6-2 所示。

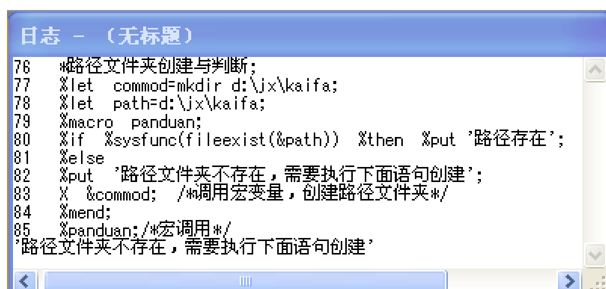


图 6-1　创建文件夹



图 6-2　文件夹路径

【程序解读】

1）%let　commod=mkdir d:\jx\kaifa：定义宏变量，赋值为 Windows 环境创建文件夹命令。

2）%let　path=d:\jx\kaifa：定义宏变量，赋值为 Windows 环境路径。

3）%if　%sysfunc(fileexist(&path))　%then　%put '路径存在'：通过宏条件语句%if 判断，fileexist(&path)函数判断宏变量 path 的值给予的路径是否存在。如果不存在输出%put 语句给出的信息到日志。如果条件不成立，执行“%else %put　'路径文件夹不存在，需要执行下面语句创建';

X　&commod;　”语句，通过 X 命令执行宏变量&commod 定义的命令。

6.3　日期处理

日期处理是 SAS 开发中对数据文件或数据集包含日期变量的数据进行处理。

【例 6.18】 读取路径 d:\jx\test 目录下的数据 data_20120601.txt、data_20120603.txt 和 data_20120601.txt。

```
/* 根据日期取相应路径 d:\jx\test 目录下的数据文件*/
%macro　filechoice(v_month,v_day,v_year); /*v_month:月份，v_day:日，v_year:年份*/
%let　path=d:\jx\test;　/*定义外部文件路径*/
%let　beg_dt=%sysfunc(mdy(&v_month,&v_day,&v_year));
%let　dt2=%sysfunc(putn(&beg_dt.,yymmdd10.));
%put　&dt2;
%let　dt3=%sysfunc(compress(&dt2.,-));
%put　&dt3;
%let　type=.txt;
%let　fil= "&path\data_&dt3&type";
%put &fil;
libname　jx　'd:\jx\test';　/*定义逻辑库*/
data　jx.jiaoyi&v_day;　/*刷卡交易一月份数据*/
```

```
        Infile   &fil dlm='|'     dsd   missover   ;
            input                 id        :$4.
                                  amt       :6.
                                  year      :$8.
                ;
        length   month $2.;
         month=substr(year,5,2);
run;
%mend   filechoice;
/*读取 20120601 数据*/
%filechoice(6,1,2012);/*调用宏过程，传递实参时间参数 v_month、v_day 和 v_year*/
/*读取 20120602 数据*/
%filechoice(6,2,2012);
/*读取 20120603 数据*/
%filechoice(6,3,2012);
```

【程序解读】

1）%macro filechoice(v_month,v_day,v_year)：定义 v_month（月份）、v_day（日）、v_year（年份）作为形参。

2）%let beg_dt=%sysfunc(mdy(&v_month,&v_day,&v_year))：生成 SAS 识别的 SAS 日期。

3）%let dt2=%sysfunc(putn(&beg_dt.,yymmdd10.))：处理日期，生成 yyyy-mm-dd 格式日期。

4) %let dt3=%sysfunc(compress(&dt2.,-))：生成 yyyymmdd 格式的日期。

5）%let   fil= "&path\data_&dt3&type"：拼接宏变量生成数据文件绝对路径文件。

【例 6.19】 对例 6.18 进行改造，传递文件路径和文件类型，生成通用程序。

```
   /* 根据日期取相应路径 d:\jx\test 目录下的数据文件*/
%macro   file_ty(v_path,v_type,v_month,v_day,v_year);
/* v_path ：文件路径，v_type：文件类型，v_month:月份，v_day:日，v_year:年份*/
%let   path=&v_path;   /*定义外部文件路径*/
%let   beg_dt=%sysfunc(mdy(&v_month,&v_day,&v_year));
%let   dt2=%sysfunc(putn(&beg_dt.,yymmdd10.));
 %put   &dt2;
 %let   dt3=%sysfunc(compress(&dt2.,-));
  %put   &dt3;
%let   type=&v_type;/*文件类型*/
%let   fil= "&path\data_&dt3&type";
  %put   &fil;
libname   jx   'd:\jx\test';   /*定义逻辑库*/
data   jx.jiaoyi&v_day;   /*刷卡交易一月份数据*/
        Infile   &fil dlm='|'     dsd   missover   ;
            input                 id          :$4.
                                amt          :6.
                                  year       :$8.
```

```
                ;
        length    month $2.;
          month=substr(year,5,2);
run;
%mend    file_ty;
/*读取 20120601 数据*/
%file_ty(d:\jx\test,.txt,6,1,2012);
/*调用宏过程，传递实参时间参数 v_path,v_type,v_month,v_day,v_year*/
/*读取 20120602 数据*/
%file_ty(d:\jx\test,.txt,6,2,2012);
/*读取 20120603 数据*/
%file_ty(d:\jx\test,.txt,6,3,2012);
```

【程序解读】

1）%macro file_ty(v_path,v_type,v_month,v_day,v_year)：定义宏形参，v_path ：文件路径，v_type：文件类型，v_month：月份，v_day：日，v_year：年份。

2）%let　path=&v_path：宏形参 v_path 外部文件路径传递给宏变量 path。

3）%let　type=&v_type：定义宏变量文件类型，宏形参 v_type 的值赋值给此宏变量。

6.4　批量读取同类文件

批量文件处理是对同一类文件通过一个 SAS 程序进行处理。实际开发中经常遇到大量的数据存储在不同的文件中，但存储的数据结构相同，只是文件分散存储。对大量的数据文件如果通过每一个文件对应一个 SAS 处理程序，这显然不符合开发需求，也不能提高读取数据文件的效率和简便性。为了程序的重复利用，就需要考虑批量读入数据文件，然后再通过数据步处理批量数据。宏过程更好地实现了程序的重复利用。

【例 6.20】 有一批交易数据文件，这些同类文件存储在“d:\jx\test”目录下，请用一个 SAS 程序实现批量读取同一类格式的文件。

```
*批量外部数据文件处理宏过程;
libname    jx "d:\jx";/*数据集存储逻辑库*/
%macro    pltras(v_file1,v_file2,v_file3,v_type);
/*  定义宏过程，并定义 3 个文件名参数和文件类型参数，传递文件名和文件类型*/
%let    type=&v_type;    /*文件格式*/
%let    fil1= d:\jx\test\&v_file1;    /*文件路径*/
%let    fil2 = d:\jx\test\&v_file2;
%let    fil3 = d:\jx\test\&v_file3;
%let    f1 = &fil1&type;    /*文件组合*/
%let    f2= &fil2&type;
%let    f3= &fil3&type;
filename    pl "(&f1,&f2,&f3)";    /*批量文件组合*/
data    jx.transinfor;
        Infile    pl    truncover    dlm='|'    lrecl=45    dsd    missover;
```

```
/*dsd 参数选项指名分隔符明显，missowver 指明缺失不覆盖*/
     length   zone            $4; /*定义变量类型和长度*/
    length   name      $8   ;
    length   mobile          $11 ;
    length   address       $10   ;
    length   credit      $10 ;
     input      zone          $ /*定义读入变量*/
               name           $
               mobile         $
               address        $
                credit        $
          ;
run;
%mend   pltras;
%pltras(trans1,trans2,trans3,.dat);
```

【程序解读】

1）%macro pltras(v_file1,v_file2,v_file3)：定义宏过程，并定义 3 个文件名参数和文件类型参数，传递文件名和文件类型。

2）%let fil1= d:\jx\test\&v_file1：文件路径组合，传递定义的形参。

3）%let f1 = &fil1&type; 文件组合，组合生成完整路径名。

4）filename pl "(&f1,&f2,&f3)"：通过 filename 语句定义批量文件组合逻辑名 pl，对应物理路径为&f1、&f2 和&f3 三个变量的物理路径值。

6.5 客户交易分析输出

实际开发中经常需要对数据文件进行处理，根据不同的条件取出不同的数据生成到对应数据集，为分析处理数据进行清洗过滤。Select 选择语句根据客户交易数据，把不同的还款级别数据输出到不同的数据集。

【例 6.21】 Select 选择语句应用，根据客户交易还款级别，把不同的还款级别数据输出到不同的数据集。

```
*外部数据文件处理宏过程;
libname   jx "d:\jx";/*数据集存储逻辑库*/
%macro   level(v_file1,v_type);
%let   type=&v_type;   /*文件格式*/
%let   fil1= d:\jx\&v_file1;   /*文件路径*/
%let   f1 = &fil1&type;   /*文件组合*/

filename   wj "(&f1)";   /*filename 定义逻辑文件名*/
data   jx.hka   jx.hkb   jx.hkc;
    Infile   wj   truncover   dlm='|'   lrecl=50   dsd   missover;
/*dsd 参数选项指名分隔符明显，missowver 指明缺失不覆盖*/
```

```
     length   zone                $4; /*定义变量类型和长度*/
   length   name               $8   ;
   length   mobile             $11;
   length   address            $10;
   length   credit             $10;
    input       zone       $ /*定义读入变量*/
                name            $
                mobile          $
                address         $
                credit          $
                level           $1.
   ;
  select   (level);    /*对 level 还款级别字段选择查询*/
     when   ('A')   output   jx.hka ;    /*条件 A 级别成立客户信息输出到 jx.hka 数据集*/
        when   ('B')   output   jx.hkb;    /*条件 B 级别成立客户信息输出到 jx.hkb 数据集*/
           otherwise   output   jx.hkc;     /*上面条件都不成立客户信息输出到 jx.hkc 数据集*/
end;
run;
%mend level;
%level(hk,.txt);/*调用宏过程，传递数据文件名和数据类型实参*/
```

【程序解读】

1）select (level)：select 语句选择条件变量 level 作为选择查询条件。

2）when ('A') output jx.hka：根据选择查询条件，level= 'A'条件成立客户信息输出到 jx.hka数据集。

3）when ('B') output jx.hkb：根据选择查询条件，level= 'B'条件成立客户信息输出到 jx.hkb数据集。

4）otherwise output jx.hkc：此还款级别分为 3 种情况，上面两种情况都不成立时客户信息输出到 jx.hkc 数据集。

6.6 批量文件压缩

对于金融业而言一般 SAS 都是在 UNIX 环境下运行，保障了系统的稳定性和效率。本节案例为 UNIX 环境下通过 SAS 程序批量压缩 UNIX 目录下按日期存储的数据文件。

【例 6.22】 批量压缩 UNIX 环境下日期为 20120231～20120401 这段日期的数据，文件路径为/home/test。文件名格式为 jy_yyyymmdd.txt。

```
options   mlogic   mprint;
%macro   plzip();
%let   path=/home/test; /*文件路径*/
%let   beg_dt=%sysfunc(mdy(2,31,2012));/*开始日期*/
%let   end_dt=%sysfunc(mdy(4,01,2012));/*结束日期*/
%do   i=&beg_dt.   %to   &end_dt.;/*日期循环*/
```

```
    %let    v_dt= %sysfunc(compress(%sysfunc(putn(&i.,yymmdd10.)),-));
/*把每次循环日期赋值给变量 v_dt*/
    %let    fil_name=jy_&v_dt.;
filename    zipping    pipe    "&path./ys.sh &path./test &fil_name. &path.";
 /*传递 3 个参数分别给
    Path _f=$1
    file_fil_name=$2
    path_tar=$3*/
data    _null_;
    infile    zipping;
    input ;
    put _infile_;
run;
%end;
%mend;

%plzip();
```

【程序解读】

1）%let beg_dt=%sysfunc(mdy(2,31,2012))：定义数据文件开始日期。

2）%let end_dt=%sysfunc(mdy(4,01,2012))：定义数据文件结束日期。

3）%do i=&beg_dt. %to &end_dt.：通过循环语句取日期。

4）%let v_dt= %sysfunc(compress(%sysfunc(putn(&i.,yymmdd10.)),-))：对日期处理，去掉日期“-”符合，把每次循环日期赋值给变量 v_dt。

5）%let fil_name=jy_&v_dt.：文件名组合，生成 jy_ yyyymmddr 格式的日期文件名。

6）filename zipping pipe "&path./ys.sh &path./test &fil_name. &path."：通过管道 pipe 执行 filename 语句中调用 ys.sh，实现压缩功能。

UNIX 环境执行压缩 SAS 程序的 SHELL 程序如下：

```
#本脚本为 UNIX 下压缩文件脚本，共需 4 个参数
#$1 xls_root  需压缩文件的路径
#$2 file_nm    需压缩文件的名字
#$3 zip_nm    压缩包名字
#$4 zip_root  压缩包存放路径

#定义变量
    root_f=$1
    file_nm=$2
    root_tar=$3
cd $root_f
tar -cvf    $file_nm.tar $file_nm
compress -f $file_nm.tar
mv    ./$file_nm.tar.Z      $root_tar
```

【程序解读】

1）cd $root_f:cd：UNIX 环境切换目录命令。

2）tar -cvf $file_nm.tar $file_nm：tar –cvf 为压缩文件命令，把$file_nm 压缩为$file_nm.tar 格式文件。

3）compress -f $file_nm.tar：压缩为.tar 格式。

4）mv ./$file_nm.tar.Z $root_tar：通过 mv 命令移动压缩好的文件到变量$root_tar 的目录。

第 7 章　统计分析基础与案例

7.1　统计分析基础

统计分析是 SAS 系统的强项，在国际上得到了广泛的应用，尤其是医药分析，以 SAS 统计分析指标作为国际标准。SAS 系统把统计学基础统计计算编译了很多常用统计过程，用户只需要知道每一个统计过程具有哪些统计指标就可以调用相应的统计过程，帮助完成对现实事物或事件的统计分析。

7.1.1　描述性统计过程概述

对事物的统计分析一般首先要对数据进行描述性统计分析，以便于描述测量样本的各种特征及其所代表的总体的特征以及发现其数据的内在规律，再选择进一步分析的方法。描述性统计分析要对调查总体所有变量的有关数据做统计性描述，主要包括数据的频数分析、数据的集中趋势分析、数据的离散程度分析、数据的分布及一些基本的统计图形。

描述性统计分析是对一组数据的各种特征进行分析，以便于描述测量样本的各种特征及其所代表的总体的特征。描述性统计分析的指标项目很多，常用的有平均数、标准差、中位数、频数分布、正态或偏态程度等，这些分析是复杂统计分析的基础。SAS 系统有 7 种常用描述性统计过程，所谓描述性统计过程是研究如何用科学的方法去搜集、整理、分析经济和社会发展的实际数据，并通过统计所特有的统计指标和指标体系，表明所研究的社会经济现象的规模、水平、速度、比例和效益，以反映社会经济现象发展规律在一定时间、地点、条件下的作用，描述社会经济现象数量之间的关系和变动规律。

理解统计学概念是应用统计分析的关键。常用统计分析概念指标如下：

（1）总体和样本

总体：根据研究目的确定的同质的研究对象的全体，更确切地说，是性质相同的所有观察单位某种变量值的集合。

样本：从总体中随机抽取的有代表性的一部分。统计分析正是通过对具体样本值的分析、研究，从而正确地推断出总体所具有的特性。这也正是统计的重要任务之一。正因为如此，对样本的来源有一定的要求，具体方法称为抽样研究。

（2）置信度、置信水平和置信区间

置信度：置信度也叫置信水平，根据来自母体的一组子样（即观测值），对表征母体的参数进行估计的统计可信程度。它是指特定个体对待特定命题真实性相信的程度。概率的置信度解释表明，事件本身并没有什么概率，事件之所以指派有概率只是指派概率的人头脑中所具有的信念证据。

置信水平：总体参数值落在样本统计值某一区内的概率。

置信区间：在某一置信水平下，样本统计值与总体参数值间的误差范围。置信区间越大，

置信水平越高。

（3）偏度和峰度

偏度和峰度是对形状测量的统计量。

偏度：偏度是指次数分布非对称的偏态方向程度。为了精确测定次数分布的偏斜状况，统计上采用偏斜度指标，描述测量值是否对称地分布在中心的两侧，分为正偏态和负偏态。

峰度：所谓峰度是指次数分布曲线顶峰的尖平程度，是次数分布的又一重要特征。统计上，常以正态分布曲线为标准。根据变量值的集中与分散程度，峰度一般可表现为 3 种形态：尖顶峰度、平顶峰度和标准峰度。变量值的次数在众数周围分布比较集中，使次数分布曲线比正态分布曲线顶峰更为隆起尖峭，称为尖顶峰度；变量值的次数在众数周围分布较为分散，使次数分布曲线较正态分布曲线更为平缓，称为平顶峰度。可见，尖顶峰度或平顶峰度都是相对正态分布曲线的标准峰度而言的。

描述性统计分析主要分为集中趋势、离散趋势和探索分析。

（1）集中趋势分析

集中趋势分析：统计学中研究一组数据向某一中心值靠拢的程度，它反映了一组数据中心点的位置所在。常用度量方法如下。

均值（mean）：表示一系列数据或统计总体的平均特征的值。

中位数（median）：将总体单位的某一数量标志的各个数值按照大小顺序排列，居于中间位置的那个数值就是中位数。

众数（mode）：变量数列中出现次数最多或频率最高的变量值。一般值的偏差比较大时，衡量统计数据指标用众数更能体现整体的特性。

（2）离散趋势

离散趋势的各测度值是对数据离散程度所做的描述，反映各变量值远离其中心值的程度，因此也称为离中趋势。它从另一个侧面说明了集中趋势测度值的代表程度。常用测量指标如下。

全距：一组变量值的最大值与最小值之差。

平均差：平均差是总体各单位标志值对其算术平均数的离差绝对值的算术平均数。

四分位差：四分位差是四分位数中间两个分位之差。

标准差：总体各单位标志值对其算术平均数离差平方的算术平均数的平方根，又称为均方差或均方根差。

方差：标准差的平方。

变异系数：标准差与平均数的比值，是衡量资料中各观测值变异程度的另一个统计量又称为离散系数，记为 C.V。

自由度：样本中可以自由变动的变量个数。如果有约束条件，自由度为样本个数减去样本数据受约束条件的个数，记为 df。

（3）探索分析

探索分析是对变量分布进行分析，了解变量分布，把握变量状态，也是统计分析需要考虑的。

SAS 系统以统计分析应用最为广泛，也是 SAS 系统的核心。SAS 系统常用统计量关键指标如表 7-1 所示。

表 7-1　SAS 系统常用统计量关键指标

常用统计量关键指标	说　明
MAX	取出一组数据的最大值
MIN	取出一组数据的最小值
N	非缺项观测记录条数
NMISS	缺项观测记录条数
RANGE	全距，为最大值与最小值之差（MAX-MIN）
SUM	求和
MEAN	算术平均值
CSS	均值校正的平方和
USS	未校正的平方和
VAR	方差
STD	标准差
STDERR	均值的标准差
CV	变异系数
SKEWNESS	偏度
KURTOSIS	峰度
T	对 H0：总体均值=0 的 student t 值
PRT	自由度 df 为 n-1 的 student t 值的双尾 P 值，为概率值
MEDIAN	中位数，按值排序，当 n 为奇数时为中间值，当 n 为偶数时为两中间值的平均数
QUARTILE	上下四分位数的值
MODE	众数，出现频率最多的值
LCML	可信区间下限
UCML	可信区间上限
ALPHA	置信水平，默认值为 0.05

SAS 分析系统常用描述性统计过程如表 7-2 所示。

表 7-2　常用描述性统计过程

描述性统计过程名	功 能 说 明
MEANS	提供一个数据汇总统计，用以对全体观测或分组观测进行描述性统计
FREQ	产生一维至 n 维的频数表和列联表，对于二维表还计算统计量并进行检验，对于 n 维表则作分层分析并在层内计算统计量
UNIVARIATE	对数值变量进行详细的描述性统计，除了提供 MEANS 过程所提供的统计描述之外，还提供变量的偏度、峰度、众数、中位数及其他的分位数等统计特征数
TABULATE	报表过程，报表方式输出统计关键字
PLOT	绘制散点图
G3D	绘制三维的曲面图。G3D 过程中使用的语句与 PLOT 过程大体相同
GPLOT	绘制连续的曲线图

7.1.2 描述性统计过程应用

为便于理解描述性统计过程实际的业界应用，本节通过案例进行讲解。

（1）MEANS 过程

MEANS 过程对数值变量进行简单的描述性统计。

【语法格式】

```
PROC      MEANS [选择项];
 VAR       变量表;
 BY        变量表;
 CLASS     变量表;
 FREQ      变量表;
 OUTPUT   [选择项];
RUN;
```

【语法解读】

1）[选择项]：为可选项，常用选项如下。

Data=：指定 SAS 数据集。

noprint：不打印输出到输出窗口。

Maxdec=：指定输出结果的最大小数位数，范围为[0,8]，为正整数，默认为 2。

可以指定统计量，常用统计量有 MAX、MIN、SUM、RANGE、CLM、CV、CSS、LCLM、N、MEAN、NMISS、STD、UCLM、VAR、SKEW、STDERR、SUMWGT、USS、PRT、T。

2）VAR：VAR 语句指定计算简单描述统计量的数值变量及次序。省略此项时，输入数据集中除 BY、CLASS、ID、FREQ 和 WEIGHT 语句中列出的变量之外所有数值变量依次都分别计算。

3）BY：BY 语句指定分组变量，组内排序后再进行统计分析。

4）CLASS：CLASS 语句与 BY 语句功能相似，只是不需要按指定变量排序。

5）FREQ：FREQ 语句指定频数分析变量。

6）OUTPUT：OUTPUT 语句指定输出分析结果到数据集。

【例 7.1】 调查某个地区 50 个人的年收入情况，求出年收入最小值、最大值、均值和全距。

```
*对外部数据处理;
%let   path= D:\jx\sr;   /*定义外部文件路径*/
%let   type=.txt;
%let   fil= "&path&type";
libname   jx 'd:\jx';   /*定义逻辑库*/
data   jx.sr;   /*数据集存储到指定逻辑库*/
      Infile   &fil   ;
            input           id   :$4.
                            year_money
                  ;
run;
```

```
proc   means   data=jx.sr   min   max   mean   range;
/*统计关键字 min、max、mean 和 range*/
   Var   year_money;   /*指定分析变量*/
run;
```

程序执行后输出窗口显示如图 7-1 所示。

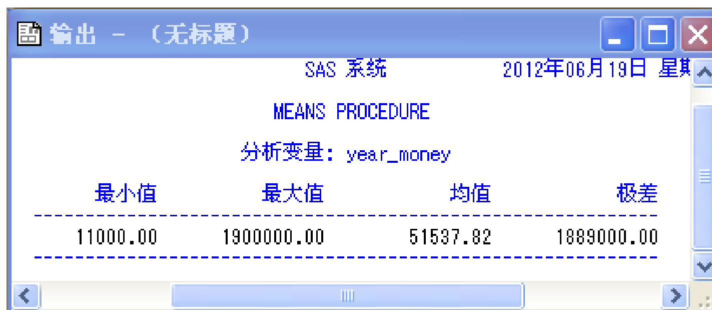


图 7-1　某地区 MEANS 过程统计分析

【程序解读】

1）PROC means data=jx.sr min max mean range：调用 MEANS 过程，指定统计关键字 MIN、MAX、MEAN 和 RANGE 分别统计分析出年收入最小值、最大值、平均值和全距。

2）var year_money：指定分析变量为年收入变量 year_money。

【分析解读】 根据 MEANS 过程统计分析输出可以看出某地区调查 50 个人，其中年收入最低为 11000，最高为 1900000，平均年收入为 51537.82，全距为 1889000。可以看出此调查数据中全距比较大，用年收入均值不能反映出某地区的年收入均值，应用众数最好。

（2）FREQ 过程

FREQ 过程产生一维至 n 维的频数表和列联表，对二维表计算统计量并进行检验，对 n 维表则作分层分析并在层内计算统计量。通过 FREQ 过程可以帮助分析变量值在数据中如何分布。

【语法格式】

```
PROC      FREQ [选择项] ;
 BY        变量表;
 TABLES     表式/选择项;
 WEIGHT      变量;
OUTPUT    [选择项];
RUN;
```

【语法解读】

1）[选择项]: 为可选项，常用选项如下。

Data=: 指定 SAS 数据集。用来说明要做 FREQ 的数据集名，如果省略这一项，则指定最新建立的数据集。

ORDER=FREQ|DATA|INTERNAL|FORMATTED: 此选项规定变量水平的记录排列次序。ORDER=FREQ 表示按频数下降的次序排列，最大的频数的水平第一个出现；ORDER=DATA 表示按输入数据集中出现的次序排列；ORDER=INTERNAL 表示按非格式化值的次序排列；

ORDER=FORMATTED 表示按格式化值的次序，默认时为 ORDER=INTERNAL。

FORMCHAR(1,2,7)=’formachar-string’（字符串）：规定构造列联表单元的轮廓线和分隔线的字符，字符串长度为 3 个字符长，1 表示垂直线，2 表示水平线，7 表示水平与垂直的交叉线。如 FORMCHAR(1,2,7)= ‘   ’（三个空格），生成表格没有轮廓线和分隔线。

PAGE：此选项要求每页只输出一张表。

NOPRINT：不打印输出到输出窗口。

2）BY：指定分组变量。

3）TABLE：作二维的列联表分析时，可使用 TABLES 语句。

4）WEIGHT：指定权重变量。

5）OUTPUT：指定输出结果到数据集。

【例 7.2】 根据地区分析刷卡交易年违约情况。

```
*对外部数据处理;
%let   path= D:\jx\weiyue;   /*定义外部文件路径*/
%let   type=.txt;
%let   fil= "&path&type";
libname   jx 'd:\jx';   /*定义逻辑库*/
data   jx.weiyue;   /*数据集存储到指定逻辑库*/
        Infile   &fil dlm='|'     dsd   missover;
                input                    diqu             :$4.
                                          year             :4.
                                          times            :5.
                                          Description      :$30.
                      ;
run;
*各区属各年伤害信用违约次数;
proc   freq   data=jx.weiyue   formchar(1,2,7)='|-+' ;
     table   year*times/ nopct ;
run;
```

程序执行后输出窗口显示如图 7-2 所示。

```
输出 - (无标题)
                          SAS 系统            2012年06月20日 星期三 上

                        FREQ PROCEDURE

                       表 - year * times

year      times

频数
行百分比
列百分比|    132|    135|    158|    178|    210|    235|  合计
--------+-------+-------+-------+-------+-------+-------+
    2011|      1|      1|      1|      1|      1|      1|     6
        |  16.67|  16.67|  16.67|  16.67|  16.67|  16.67|
        |  50.00|  50.00|  50.00|  50.00|  50.00|  50.00|
--------+-------+-------+-------+-------+-------+-------+
    2012|      1|      1|      1|      1|      1|      1|     6
        |  16.67|  16.67|  16.67|  16.67|  16.67|  16.67|
        |  50.00|  50.00|  50.00|  50.00|  50.00|  50.00|
--------+-------+-------+-------+-------+-------+-------+
合计            2       2       2       2       2       2     12
```

图 7-2 刷卡违约情况分析

【程序解读】

1）proc freq data=jx.weiyue formchar(1,2,7)='|-+'：调用 FREQ 过程，DATA=jx.weiyue 指定分析数据集，formchar(1,2,7)='|-+'指定垂直线为|，水平线为-，水平与垂直的交叉线为+。

2）table year*times/ nopct：table 语句指定绘制二维列联表，year 为行，times 为列，nopct 求百分比。

（3）UNIVARIATE 过程

UNIVARIATE 过程的作用是对数值变量进行详细的描述性统计，除了提供 MEANS 过程所提供的统计描述之外，还提供变量的偏度、峰度、众数、中位数及其他的分位数等统计特征数。

【语法格式】

```
PROC      UNIVARIATE [选择项];
  VAR       变量表;
  BY        变量表;
  FREQ      变量;
  WEIGHT      变量;
  ID  变量表;
  OUTPUT   [选择项];
RUN;
```

【语法解读】

[选择项]：为可选项，常用选项如下。

1）data=：指定 SAS 数据集，如果省略这一项，则指定最新建立的数据集。

2）noprint：不打印输出到输出窗口。

3）Vardef=df|weight|wgt|n|wdf：方差计算中规定除数，df 表示除数使用自由度（n−1），为默认值。weight|wgt 表示用权数和作为除数。n 表示观测个体数（样本含量）做除数。wdf 表示用权数和减 1 做除数。

4）Freq：要求生产包含变量值、频数、百分数和累计频数的频数表。

5）Normal：要求计算关于输入数据服从正态分布的假设的检验统计量。

6）plot：生成茎叶图，一个盒形图和一个正态概率图。

7）Pctldef=：规定计算百分位数方法，取值为 1、2、3、4 和 5。

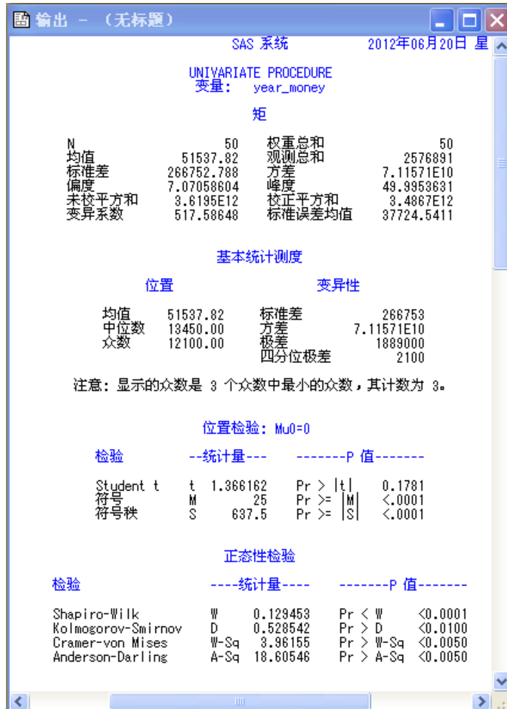
8）Round=：指定变量数值四舍五入的单位。

【例 7.3】 对例 7.1 进行改造，通过 UNIVARIATE 分析。

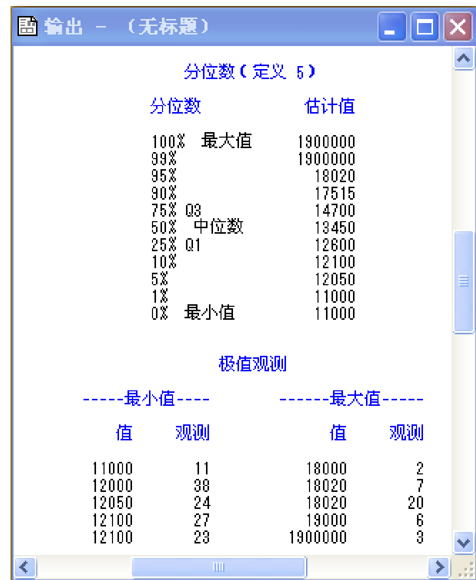
```
*对外部数据处理;
%let   path= D:\jx\sr;   /*定义外部文件路径*/
%let   type=.txt;
%let   fil= "&path&type";
libname   jx   'd:\jx';   /*定义逻辑库*/
data   jx.sr;   /*数据集存储到指定逻辑库*/
            Infile   &fil   ;
             input             id                  :$4.
                         year_money
                  ;
```

```
run;
proc   print   data=jx.sr;
run;
proc   univariate   data=jx.sr   freq   normal   plot;
   var   year_money;   /*指定分析变量*/
run;
```

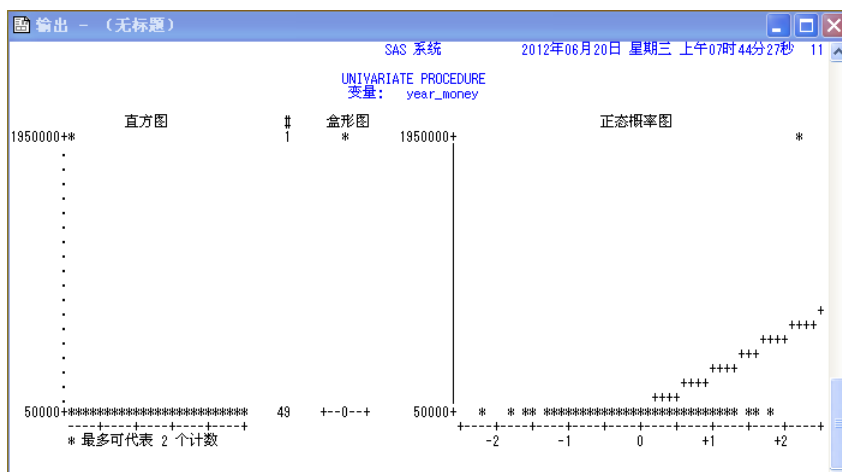
程序执行后输出窗口部分截图显示如图 7-3 所示。



过程输出（基本分析信息）



过程输出（分位数和极值观测）



过程输出（直方图和正态概念图）

图 7-3

【程序解读】

1）PROC univariate data=jx.sr freq normal plot：调用 UNIVARIATE 过程，对 data=选项指定数据集为 jx.sr，freq normal plot 分别表示求频数、正态分别和绘制直方图、盒形图和正态概率图。

2）var year_money：指定分析变量为年收入 year_money。

（4）TABULATE 过程

TABULATE 过程是一个报表过程，以报表方式输出描述性统计量。报表最多为三维：列、行和项。

一个最简单的 PROC TABULATE 过程必须定义下列三要素：

1）类变量——可以是数值型变量或字符型变量。分类的目的是为了在每一个类上进行计算和分析。

2）分析变量—— 一定是数值型变量。可以计算的一些统计量如：频数（FREQUENCY）、均值（MEAN）、标准差（STANDARD DEVIATION）、最小值（MINIMUM）、最大值（MAXIMUM）、极差（RANGE）、总和（SUM）、百分数（PERCENTAGES）等。

3）表的结构和格式——最多可以定义 3 个维度：第一维定义列，第二维定义行，第三维定义项，并且可以通过 TABLE 语句中的表达式计算统计量，用操作符（如逗号、空格、星号、圆括号等）来组织单元集合。另外，还能格式化单元中的数据值和产生用户化的行标题和列标题。

【语法格式】

```
PROC TABULATE   <选项列表> ;
     CLASS         分类变量列表 ;
     VAR           分析变量列表 ;
     TABLE        <<页表达式，>行表达式，>列表达式</表选项> ;
     LABEL        变量 1=“标签*内容”  变量 2=“标签*内容”…… ;
     KEYLABEL   统计量名字 1=“标记 1”  统计量名字 2=“标记 2”…… ;
     FORMAT       变量输出格式 ;
     WHERE          条件表达式 ;
Run ;
```

【语法解读】

1）<选项列表>：为可选项，常用选项如下。

data=：指定 SAS 数据集，如果省略这一项，则指定最新建立的数据集。

missing：此选项要求把缺失值作为分类变量的有效水平。

Vardef=df|weight|wgt|n|wdf：规定用于计算方差的除数。

Format=格式名：指定每个报表单元默认的输出格式。

ORDER=FREQ|DATA|INTERNAL|FORMATTED：此选项规定报表中分类变量值出现的次序，如省略该项，按内部值的次序。

FORMCHAR(index-list)=’string’：规定构造报表轮廓线和分隔线的字符。如 FORMCHAR (3 4 10 11)=‘* - - -   ’（3 个空格），生成的表格没有轮廓线和分隔线。

noseps：要求在表体中不出现水平分隔线。

2）CLASS：指明输入数据集中哪些变量作为分类变量。

3）VAR：指明输入数据集中哪些变量作为分析变量。

4）TABLE：指明绘制表格表达式。

5）LABEL：对分类变量或分析变量指定标签文字。

6）KEYLABEL：与 LABEL 语句差不多，是对统计量给予指定标签描述。

7）FORMAT：对分类变量指定输出显示格式。

8）WHERE：指定对数据集处理的过滤条件。

【例 7.4】 通过 TABULATE 过程分析消费支出调查数据。

```
*对外部数据处理;
%let   path= D:\jx\xiaofei;   /*定义外部文件路径*/
%let   type=.txt;
%let   fil= "&path&type";
libname   jx 'd:\jx';   /*定义逻辑库*/
data   jx.xiaofei;   /*数据集存储到指定逻辑库*/
     Infile   &fil   dlm='|'   dsd   missover;
          input                      id            :$4.
                                      sr            :6.
                                      xiaofei       :5.
                                      education     :4.
                                      Suidian       :4.
                 ;
run;
*消费支出调查分析;
proc   tabulate   data=jx.xiaofei   formchar='|----|+|---'   format=7.2 ;
   class     id ;
      var        sr    xiaofei    education    suidian ;
       table    (id all),
       (sr*mean*f=7. xiaofei*MAX education*MIN suidian*MEAN ) /rts=8 ;
       Keylabel     MEAN="平均值"    all="月收入" ;
       Label      suidian='月水电花费' ;
      Title1    '消费支出某月调查分析' ;
      Title2    '年份  2012' ;
run ;
```

程序执行后输入窗口显示如图 7-4 所示。

【程序解读】

1）formchar='|----|+|---'：指定报表表格水平线、分隔线和交叉线。

2）format=7.2：指定数据输出显示格式。

3）class id：指定按 id 分类。

4）var sr xiaofei education suidian：指定分析变量为 sr、xiaofei、education 和 suidian。

5）table (id all),(sr*mean*f=7. xiaofei*MAX education*MIN suidian*MEAN) /rts=8：绘制表格，根据 id，绘制 all 表示所有，求 sr 的均值，其他设置属性类似。

6）Keylabel MEAN="平均值" all="月收入"：指定标签描述。

7）Title1 和 Title2 指定标题。

输出 - （无标题）

消费支出某月调查分析　　2012年06月21日

年份 2012

	sr	xiaofei	educat-ion	月水电花费
	平均值	Max	Min	平均值
id				
1001	4500	2300.00	1500.00	120.00
1002	6700	3200.00	2500.00	280.00
1003	5200	3100.00	1800.00	220.00
1004	8500	4628.00	1600.00	320.00
1005	12500	6300.00	2100.00	250.00
1006	8900	5305.00	1900.00	150.00
1007	9300	5100.00	2400.00	160.00
1008	3900	1000.00	500.00	50.00
月收入	7438	6300.00	500.00	193.75

图 7-4　消费支出分析

（5）PLOT 过程

PLOT 过程的作用是绘制散点图，此过程帮助分析数据变化的趋势、数据间的相关性。在 PROC PLOT 语句中的绘图表达式为 y*x，y 表示纵坐标、x 表示横坐标。一般情况下，PLOT 过程用字符 A 表示点，两个点重合时用字符 B 表示，3 个点重合时用字符 C 表示。也可以指定语句表示：

```
PLOT　A*B=‘*’（用*表示点）;
PLOT　Y*X=变量 （用变量的值表示点）;
```

在 PLOT 语句中可以使用下列选择项：

1）VAXIS=数值，指定纵轴上的刻度标记。

2）HAXIS=数值，指定横轴上的刻度标记。

3）OVERAY，指定将所有的图组合输出，相互重叠覆盖。

【语法格式】

```
PROC PLOT [选择项];
    BY  变量表;
    PLOT  表式/选择项;
RUN;
```

【语法解读】

1）[选择项]：为可选项。常用选项如下。

① data=：指定 SAS 数据集，如果省略这一项，则指定最新建立的数据集。

② VTOH=数值：指定 PLOT 过程输出时纵横坐标轴的比例。

③ VPERCENT=数值：指定 PLOT 过程产生图表长度所占一页长度的纵向百分比的分子。例如，VPERCENT=33 指定 PLOT 过程在一页上垂直输出 3 个图，每个图各占一页长的 1/3；VPERCENT=33 0 指定 PLOT 过程在一页上垂直输出的图仅占一页长的 1/3，但这一页只输出这一个图。

④ HPERCENT=数值：指定 PLOT 过程产生图表宽度所占一页长度的纵向百分比的分子，用法同 VPERCENT。

2）BY 变量表：指定散点图按 BY 语句指定变量次序分类。

3）PLOT 表式/选择项：指定绘图表达式。

【例 7.5】 对消费支出调查数据绘制散点图。

```
*对外部数据处理;
%let   path= D:\jx\xiaofei;   /*定义外部文件路径*/
%let   type=.txt;
%let   fil= "&path&type";
libname   jx 'd:\jx';   /*定义逻辑库*/
data   jx.xiaofei;   /*数据集存储到指定逻辑库*/
            Infile   &fil   dlm='|'   dsd   missover;
             input                  id          :$4.
                                    sr          :6.
                                    xiaofei     :5.
                                    Education   :4.
                                    Suidian     :4.
                 ;
run;
*消费支出调查分析散点图;
proc   plot   data=jx.xiaofei   hpercent=80;
    Plot   sr*xiaofei='*';
run;
```

程序执行后输出窗口显示如图 7-5 所示。

【程序解读】

1）hpercent=80：用来指定 PLOT 过程产生图表宽度所占一页长度的纵向百分比的分子。

2）plot sr*xiaofei='*'：指定纵轴为 sr，横轴为 xiaofei，用*表示点。

（6）G3D 过程

G3D 过程的作用是绘制三维的曲面图，G3D 过程中使用的语句与 PLOT 过程大体相同。

【语法格式】

```
PROC G3D [选择项];
     BY  变量表;
     PLOT  表式/选择项;
RUN;
```

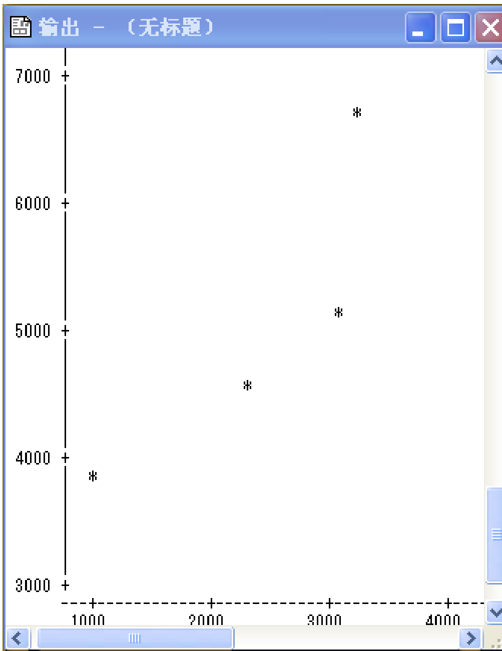


图 7-5　消费分析散点图

【例 7.6】 绘制 y=kx+3 的图像，x 从–6～6，以 0.5 递增，k 从–6～6，以 0.5 递增。

```
data  ex;
   do  x=-6  to  6  by  0.5;
     do  k=-6  to  6  by  0.5;
     y=k*x+3;
     Output;
     end;
  end;
proc  g3d;
  Plot  k*x=y;
run;
```

程序执行后图像窗口显示如图 7-6 所示。

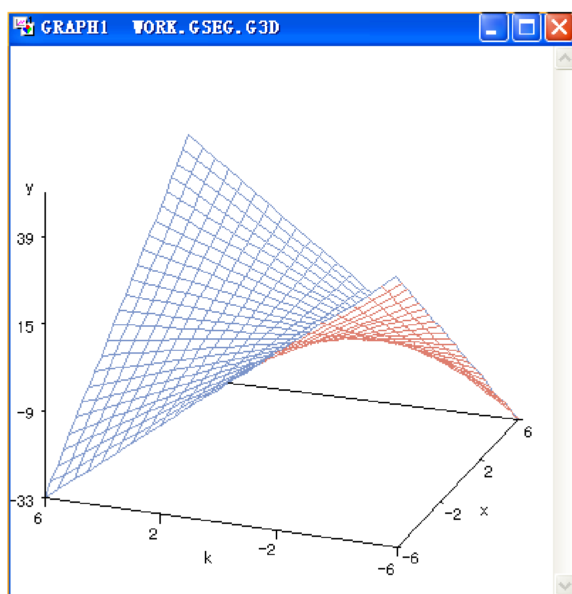


图 7-6　G3D 过程图形显示

（7）GPLOT 过程

GPLOT 过程的作用是绘制连续的曲线图。GPLOT 过程中使用的语句与 PLOT 过程大体相同，用到时可以参考。

【语法格式】

```
PROC GPLOT [选择项];
    BY  变量表;
    PLOT  表式/选择项;
RUN;
```

【例 7.7】 绘制余弦图和一元直线图。

```
data  gplot;
```

```
 do   ds=0   to 360;
    y=cos(ds*3.1415926/180);
     k=2*ds/180-2;
    Output;
end
 symbol   i=spline;
proc   gplot;
  plot   y*ds k*ds/overlay;
run;
```

程序执行后输出结果显示如图 7-7 所示。

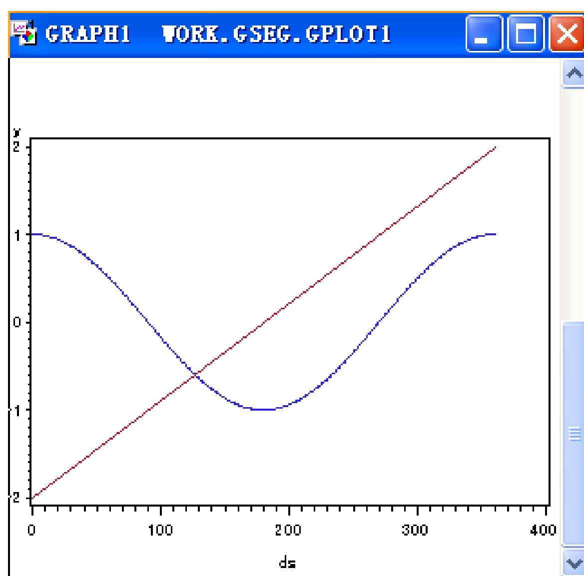


图 7-7　余弦图

【程序解读】

plot y*ds k*ds/overlay：分别绘制 y 与 ds 的关系图以及 k 与 ds 的关系图。

7.2　方差分析基础

方差分析（Analysis of Variance，ANOVA）又称“变异数分析”或“F 检验”，是 R. A. Fisher 发明的，用于两个及两个以上样本均数差别的显著性检验。由于各种因素的影响，研究所得的数据呈现波动状。方差分析是检验两个或两个以上样本均数差异是否具有统计意义的一种方法。

7.2.1　方差分析概述

方差分析是从观测变量的方差入手，研究诸多控制变量中哪些变量是对观测变量有显著影响的变量。事物往往有许多因素互相制约又互相依存，方差分析的目的是通过数据分析找出对该事物有显著影响的因素，各因素之间的交互作用，以及显著影响因素的最佳水平等。

方差分析是在可比较的数组中，把数据间的总的“变差”按各指定的变差来源进行分解的一种技术。对变差的度量，采用离差平方和。方差分析方法就是从总离差平方和分解出可追溯到指定来源的部分离差平方和。经过方差分析若拒绝了检验假设，只能说明多个样本总体均数不相等或不全相等。若要得到各组均数间更详细的信息，应在方差分析的基础上进行多个样本均数的两两比较。

方差分析的假定条件如下：

1）各处理条件下的样本是随机的。

2）各处理条件下的样本是相互独立的，否则可能出现无法解析的输出结果。

3）各处理条件下的样本分别来自正态分布总体，否则使用非参数分析。

4）各处理条件下的样本方差相同，即具有齐效性。

方差分析的假设检验：

假设有 K 个样本，如果原假设 H0 样本均数都相同，K 个样本有共同的方差 σ，则 K 个样本来自具有共同方差 σ 和相同均数的总体。如果经过计算，组间均方远远大于组内均方，则推翻原假设，说明样本来自不同的正态总体，说明处理造成均值的差异有统计意义。否则承认原假设，样本来自相同总体，处理间无差异。

（1）单因素方差分析

单因素方差分析是用来研究一个控制变量的不同水平是否对观测变量产生了显著影响。这里，由于仅研究单个因素对观测变量的影响，因此称为单因素方差分析。例如，分析不同经济指标带来显著影响，考查经济差异是否影响教育水平等。这些问题都可以通过单因素方差分析得到答案。

单因素方差分析的第一步是明确观测变量和控制变量。单因素方差分析的第二步是剖析观测变量的方差。方差分析认为：观测变量值的变动会受控制变量和随机变量两方面的影响。据此，单因素方差分析将观测变量总的离差平方和分解为组间离差平方和和组内离差平方和两部分，用数学形式表述为 SST=SSA+SSE。单因素方差分析的第三步是通过比较观测变量总离差平方和各部分所占的比例，推断控制变量是否给观测变量带来了显著影响。

单因素方差分析原理：在观测变量总离差平方和中，如果组间离差平方和所占比例较大，则说明观测变量的变动主要是由控制变量引起的，可以由控制变量来解释，控制变量给观测变量带来了显著影响；反之，如果组间离差平方和所占比例小，则说明观测变量的变动不是由控制变量引起的，不可以由控制变量来解释，控制变量的不同水平没有给观测变量带来显著影响，观测变量值的变动是由随机变量因素引起的。

单因素方差分析的基本步骤如下。

1）出原假设：H0——无差异，H1——有显著差异。

2）选择检验统计量：方差分析采用的检验统计量是 F 统计量，即 F 值检验。

3）计算检验统计量的观测值和概率 P 值：该步骤的目的就是计算检验统计量的观测值和相应的概率 P 值。

4）给定显著性水平，并作出决策。

单因素方差分析在完成上述单因素方差分析的基本分析后，可得到关于控制变量是否对观测变量造成显著影响的结论，接下来还应做其他几个重要分析，主要包括方差齐性检验、多重比较检验。

① 方差齐性检验是对控制变量不同水平下各观测变量总体方差是否相等进行检验。前面提到，控制变量不同各水平下观测变量总体方差无显著差异是方差分析的前提要求。如果没有满足这个前提要求，就不能认为各总体分布相同。因此，有必要对方差是否齐性进行检验。

② 多重比较检验单因素方差分析的基本分析只能判断控制变量是否对观测变量产生了显著影响。如果控制变量确实对观测变量产生了显著影响，还应进一步确定控制变量的不同水平对观测变量的影响程度如何，其中哪个水平的作用明显区别于其他水平，哪个水平的作用是不显著的等。

检验统计量的构造方法：LSD 方法和 S-N-K 方法。

LSD 方法称为最小显著性差异（Least Significant Difference）法。最小显著性差异法即水平间的均值只要存在一定程度的微小差异就可能被检验出来。它利用全部观测变量值，而非仅使用某两组的数据。LSD 方法适用于各总体方差相等的情况，但它并没有对犯一类错误的概率问题加以有效控制。

S-N-K 方法：S-N-K 方法是一种有效划分相似性子集的方法。该方法适合于各水平观测值个数相等的情况。

（2）多因素方差分析

多因素方差分析用来研究两个及两个以上控制变量是否对观测变量产生显著影响。这里，由于研究多个因素对观测变量的影响，因此称为多因素方差分析。多因素方差分析不仅能够分析多个因素对观测变量的独立影响，更能够分析多个控制因素的交互作用能否对观测变量的分布产生显著影响，进而最终找到利于观测变量的最优组合。

（3）协方差分析

协方差分析是把直线回归与方差分析法结合起来的一种方法，利用回归的关系消除对比各组自变量值不同所产生的影响后，再进行方差分析。

协方差分析将那些人为很难控制的控制因素作为协变量，并在排除协变量对观测变量影响的条件下，分析控制变量（可控）对观测变量的作用，从而更加准确地对控制因素进行评价。协方差分析仍然沿承方差分析的基本思想，并在分析观测变量变差时，考虑了协变量的影响。人为观测变量的变动受 4 个方面的影响（控制变量的独立作用、控制变量的交互作用、协变量的作用和随机因素的作用），在扣除协变量的影响后，再分析控制变量的影响。

方差分析中的原假设是：协变量对观测变量的线性影响是不显著的；在协变量影响扣除的条件下，控制变量各水平下观测变量的总体均值无显著差异，控制变量各水平对观测变量的效应同时为零。检验统计量仍采用 F 统计量，它们是各均方与随机因素引起的均方比。

SAS 系统中 ANOVA 过程是进行方差分析的内部过程，此过程是一个交互方式执行，执行完 RUN 语句后该过程并没有结束。如果想结束此过程，则后面必须使用 QUIT 语句。

【语法格式】

```
PROC ANOVA < options > ;
    CLASS variables < / option > ;
    MODEL dependents=effects < / options > ;
```

```
        BY variables ;
        MEANS effects < / options > ;
        TEST < H=effects > E=effect ;
        MANOVA H=effects E=effects[/option];
RUN;
QUIT;
```

【语法解读】

1）CLASS variables < / option >：指定分类变量。

2）MODEL dependents=effects < / options >：定义拟合模型，因变量=效应表[/选择项]。

3）BY variables：指定分组变量表。

4）MEANS effects < / options >：在 MODEL 语句后可以使用多个 MEANS 语句，计算和比较均值。

5）TEST < H=effects > E=effect：方差分析中利用其他效应作为误差项，指定效应平方和和误差项，构建检验。

6）MANOVA H=effects E=effects[/option]：做多元方差分析。H=效应表，指定一个或多个需分析的效应作为假设检验矩阵。E=效应，指定产生误差项的效应。缺项时用剩余平方和、交叉积和残差矩阵作为误差项。

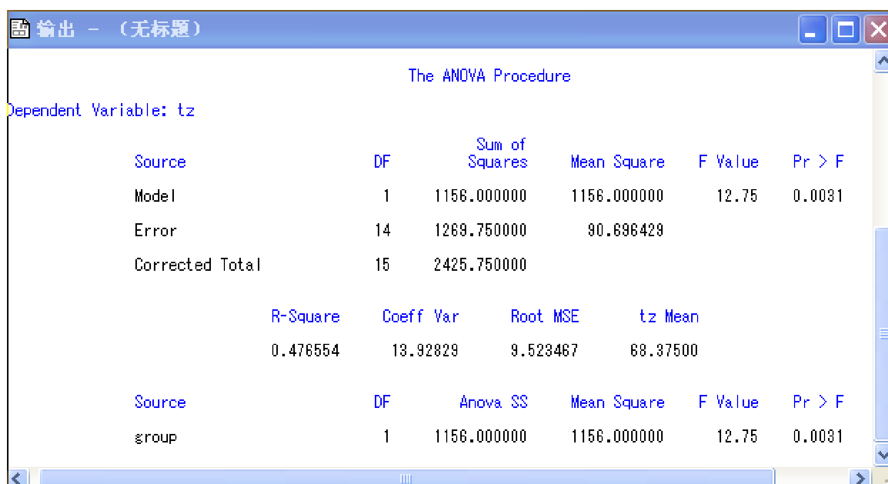
7.2.2 方差分析应用

方差分析通常应用在自变量与因变量的相关关系研究中，是看某个因素对因变量是不是有显著性的影响，主要分析各效应的显著性。

【例 7.8】 研究正常人体重与非正常人体重的差异，数据保存在 d:\jx\sg_tz.txt 文本中。

```
    *对外部数据处理;
%let   path= D:\jx\sg_tz;   /*定义外部文件路径*/
%let   type=.txt;
%let   fil= "&path&type";
libname   jx 'd:\jx';   /*定义逻辑库*/
data   jx.sg_tz;   /*数据集存储到指定逻辑库*/
        Infile   &fil   dlm='|'   dsd   missover;
                input                    group   :1.
                                          sg      :3.
                                          tz       :8.
                    ;
run;
/*调 anova 过程分析*/
proc   anova   data=jx.sg_tz;
    class   group;
    model   tz=group;
run;
```

程序执行后输出窗口显示如图 7-8 所示。



输出 - (无标题)

The ANOVA Procedure

Dependent Variable: tz

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	1	1156.000000	1156.000000	12.75	0.0031
Error	14	1269.750000	90.696429		
Corrected Total	15	2425.750000			

R-Square	Coeff Var	Root MSE	tz Mean
0.476554	13.92829	9.523467	68.37500

Source	DF	Anova SS	Mean Square	F Value	Pr > F
group	1	1156.000000	1156.000000	12.75	0.0031

图 7-8　体重分析（一）

【程序解读】

1）class group：指定分组变量。

2）model tz=group：建立模型，体重=两组之间方差分析。

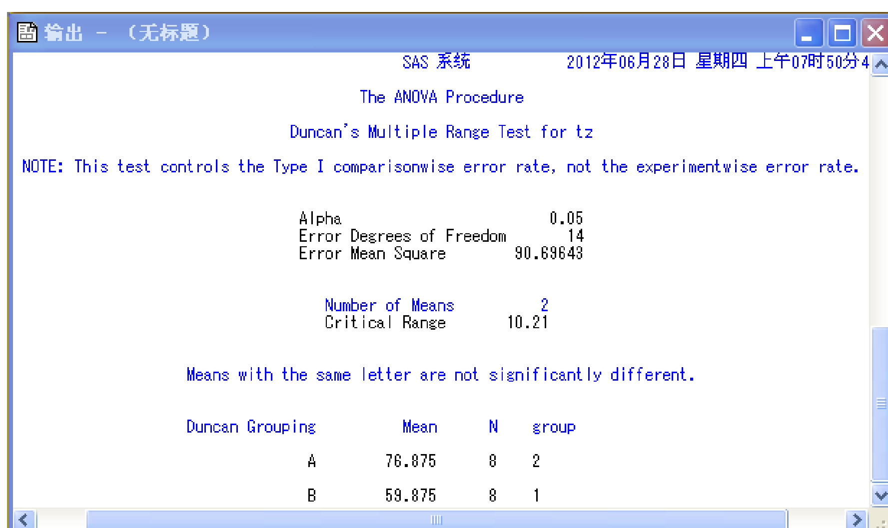
【分析结果解读】

看变量 group 的 F 值，F=12.75, P=0.0031<0.01，可以得出两组体重有显著性差异。

为了知道研究两组之间的差异是否有意义，需做 Q 检验，在上面的程序运行后，执行如下程序：

```
means group/duncan;
RUN;
QUIT;
```

程序执行后输出窗口显示如图 7-9 所示。



输出 - (无标题)

SAS 系统　　2012年06月28日 星期四 上午07时50分4

The ANOVA Procedure

Duncan's Multiple Range Test for tz

NOTE: This test controls the Type I comparisonwise error rate, not the experimentwise error rate.

Alpha	0.05
Error Degrees of Freedom	14
Error Mean Square	90.69643

Number of Means	2
Critical Range	10.21

Means with the same letter are not significantly different.

Duncan Grouping	Mean	N	group
A	76.875	8	2
B	59.875	8	1

图 7-9　体重分析（二）

【分析结果解读】

Error Mean Square 为误差均方，number of means 和 Critical Range 分别表示比较的组数和它们之间相差有无意义的临界值。

7.3 相关分析与回归分析基础

相关分析是研究变量之间相互关系的密切程度和相互联系方式的重要方法。对于描述随机现象的变量，它们之间的关系具有某种不确定性，通过相关分析与回归分析研究实际事物与现象之间的关系。

7.3.1 相关分析与回归分析概述

1．相关分析的概念

相关分析是为了检验分析变量之间是否存在某种联系，以及变量之间联系的密切程度，其联系的密切程度通过相关系数衡量。

相关的函数关系是一种严格的依存关系，这种关系可以用 $y = f(x)$来表现。

相关的密切程度关系系数分为正负号，分别表示正相关和负相关。其 R 值的取值范围在[0,1]绝对值之间。其值的大小反映两变量之间相关的密切程度，如表 7−3 所示。

表 7−3 变量之间相关系数 R 的强度说明

R 值	相 关 强 度
1	完全正相关
−1	完全负相关
0	完全不相关
0.8 以上	非常强相关
0.60～0.80	强相关
0.40～0.60	中等相关
0.20～0.40	弱相关
0.20 以下	非常弱相关

2．回归分析的概念与分类

回归分析是对具有相关关系的两个或两个以上变量之间的数量依存关系进行测定，确立一个相应的数学表达式，以便从一个已知量来推测另一个未知量，为估算预测提供一个重要的方法。通过自变量的变化预测因变量的变化趋势。

回归的分类：按自变量的个数分为一元回归和多元回归，按回归线的形状分为线性回归和非线性回归。

简单线性回归方程式的一般形式：$y=a+bx$。

【说明】 所分析的两个变量之间必须存在相关关系，且相关程度在显著相关以上。当两个变量的增长比例为常数时，它们之间就呈现为一种简单线性关系。

对于两变量相关的散点图中，引出一条最优的直线，这条直线就是估计回归线。它表明了两个变量数量变动的一般关系，估计回归方程如下：

$$y_c = a + bx$$

说明：

y_c：因变量估计值。

a：直线截距。

b：直线的斜率，又称回归系数。

x：自变量

3．相关分析过程CORR过程

SAS 系统 CORR 过程是研究变量之间相关性的过程，通过此过程看变量之间的相关程度。尤其是在风险控制和医药行业经常用此过程研究变量之间的相关性，以便于发现显著因素，从而控制某几个变量。

【语法格式】

```
PROC   CORR <选项>;
          BY   变量列表;
          FREQ   变量;
          PARTIAL  变量列表;
          VAR   变量列表;
                  WEIGHT  变量;
                  WITH  变量列表;
              RUN;
```

【语法解读】

1）CORR：过程名，实现变量相关程度分析的 SAS 内部过程。

2）<选项>：可选项。常用选项如表 7−4～表 7−6 所示。

表 7−4　CORR 过程常用数据集选项设置说明

选　　项	说　　明
DATA=	指定要分析的数据集，省略时默认当前最新数据集
OUTH=	输出为 Hoeffding's D 统计分析输出
OUTK=	Kendall 相关统计分析输出
OUTP=	Pearson（皮尔森）相关统计分析输出
OUTS=	Spearman 相关统计分析输出

对于 CORR 过程可以对统计分析设置控制统计分析参数选项，如表 7−5 所示。

表 7−5　控制统计分析参数选项

选　　项	说　　明
EXLNPWGT	排除负数权重值的观测记录
NOMISS	排除缺失值的观测记录
PEARSON	要求皮尔森积矩相关

对于 CORR 过程，统计分析过程中可以对分析显示信息进行打印输出控制。控制打印输出选项说明，如表 7−6 所示。

表 7-6　控制打印输出选项说明

选　项	说　明
BEST=	指定打印 N 的高度相关，N>=1，顺序从高到低
NOCORR	抑制 Pearson 相关显示
NOPRINT	抑制所有显示输出，只是创建数据集
NOPROB	抑制每个相关的概率性显示
NOSIMPLE	抑制对每个变量的简单描述统计的打印信息
RANK	指定显示按排序的相关系数显示

3）Pearson（皮尔森）相关统计的参数如下。

ALPHA：计算 ALPHA 值。

COV：计算协方差。

NOMISS：排除缺失值。

4）PARTIAL　v_var1 语句：对变量进行偏相关统计。

5）WITH　v_var1 语句：和 VAR 变量组合应用，用来指定矩阵的横轴。

4．回归分析过程REG过程和LOGISTIC过程

（1）REG 回归过程

REG 回归过程是 SAS 系统中的一个比较通用的普通过程，不像 LOGISTIC（逻辑回归过程）、NLIN（非线性回归过程）和 CATMOD（分类过程）等过程具体适用于某一类。REG 过程适用的回归分析比较通用，没有太多限制。

【语法格式】

```
PROC   REG  < 选项 > ;
            < label: >   MODEL dependents=<regressors> < / options > ;
            BY   v_variable ;
            FREQ   v_variable ;
            ID   v_variables ;
            VAR   v_variables ;
            WEIGHT   v_variable ;
            ADD   v_variable ;
            DELETE   v_variable;
            < label: > MTEST <equation, ... ,equation> < / options > ;
            OUTPUT < OUT=SAS-data-set > keyword=names
                               < ... keyword=names > ;
            PLOT <y_variable*x_variable> <=symbol>
                        < ...yvariable*xvariable> <=symbol> < / options > ;
            < label: > TEST equation,<, ...,equation> < / option > ;
         RUN;
```

【语法解读】

1）REG：REG 回归过程的调用，是一个普遍应用的回归过程。

2）REG 回归可以加入一些选项参数。常用的选项如表 7-7 所示。

表 7-7 REG 过程常用选项说明

选　　项	说　　明
DATA=	指定回归分析的数据集名，默认取当前最新数据集
OUTEST=	指定回归分析中包含参数估计和模型的一些统计信息的数据集名，也就是把包含参数估计和模型统计的一些信息存储到此处指定的数据集保存
OUTSSCP=	指定一个数据集的名字，存储 SCP 类型的相关矩阵的分析信息，包含如平方和和变量交叉等分析信息。对于大数据量的分析，可以把此类结果存储到指定数据集保存分析信息，以方便分析再利用，而不是输出到输出窗口
COVOUT	此选项参数是在有 OUTEST=选项时才可以应用，把参数估计的协方差矩阵输出到此数据集
OUTSTB	此选项参数是输出标准的参数估计信息到 OUTEST=选项所指定的数据集中
CORR	对 MODEL 语句或 VAR 语句声明的变量显示所列举变量的相关系数矩阵
SIMPLE	此选项对 MODEL 语句或 VAR 语句指定的变量显示基本的统计信息，如均值、方差和标准差等
USCCP	显示未修正平方和或交叉矩阵
ALL	此选项表示显示所有的统计信息
NOPRINT	对分析结果不输出到输出窗口

3）常用过程语句说明如下。

< label: >　MODEL dependents=<regressors> < / options >：< label: >为可选项，为模型指定标号；dependents=<regressors>指定因变量=自变量，这是对分析数据集建立模型时指定研究变量与变量之间的关系模型。

< label: > MTEST <equation, ... ,equation> < / options >：自定义回归参数假设的等式，多个等式之间以逗号分隔。

OUTPUT < OUT=SAS-data-set > keyword=names < ... keyword=names >：OUTPUT 语句指定把回归分析的分析结果存储到指定的数据集，OUT=指定存储到的数据集名字；keyword=names 指定分析输出到指定数据集的关键分析指标。

< label: > TEST equation,<, ...,equation> < / option >：和 MTEST 语句用法一样。

PLOT <y_variable*x_variable> <=symbol> < ...yvariable*xvariable> <=symbol> < / options >：PLOT 语句指定绘制散点图，所取的绘图变量需要从 MODEL 语句或 VAR 语句中选择; <y_variable*x_variable>指定绘图的 Y 轴变量和 X 轴变量；<=symbol>指定图形绘制中的表示符号。可以绘制多个散点图形。

（2）LOGISTIC 逻辑回归过程

LOGISTIC 逻辑回归过程是一个二值响应回归过程，也就是分析的变量分类只有两个值 0 和 1 或次序变量的关系。LOGISTIC 回归属于一种特殊的回归过程，用时要根据需求来定。如果此事件分析变量具有二值类型，只有两种结果，就可以选择 LOGISTIC 逻辑回归模型进行分析。数据挖掘中二值事件的情况经常用到逻辑回归模型。

LOGISTIC 逻辑回归经常用来做二值响应模型的分析，是一种特殊的回归过程。事件为二值类型的可以选择此回归方式。

语法格式：PROC LOGISTIC < 选项>;

```
BY   v_variables;
CLASS   v_variable <(v-options)> <v_variable <(v-options)>... >
   < / v-options >;
FREQ   v_variable ;
```

```
MODEL events/trials = < effects > < / options >;
MODEL v_variable < (variable_options) > = < effects > < / options >;
OUTPUT < OUT=SAS-data-set > < keyword=name...keyword=name > /
        < option >;
< label: > TEST equation1 < , ... , < equationk >> < /option >;
WEIGHT    v_variable </ option >;
RUN;
```

【语法解读】

1）LOGISTIC 过程中 PROC LOGISTIC 语句和 MODEL 语句是必须选的语句，调用此过程时只有一个模型被指定，其余都是可选择语句。

2）LOGISTIC 过程具有的常用选项如表 7-8 所示。

表 7-8　LOGISTIC 过程常用选项说明

选　　项	说　　明
DATA=	指定分析的数据集名
DESCENDING	对分析输出降序排序，简写为 DESC
COVOUT	此选项与 OUTEST=一起使用才有效，把估测协方差矩阵信息输出到 OUTEST=指定的数据集中
NOSIMPLE	对自变量描述性统计信息不打印输出
NOPRINT	不打印输出结果到输出窗口

3）FREQ：指定求变量的频度。

4）MODEL：MODEL 语句是定义 LOGISTIC 过程中的模型。这两类模型在实际应用中要根据需求选择，只能取其一。

① MODEL events/trials = < effects > < / options >：此模型为事件/试验模型，只应用在二分类数据。用“/”分隔事件和试验。

② MODEL v_variable < (variable_options) > = < effects > < / options >：此模型为单实验模型，主要应用在二分类数据和有序分类数据。等号前指定因变量。

以上两类模型常用选择项如下：

LINK=：指定对于线性预测的响应概率的联系函数，可以简写为 L=，默认 LINK=LOGIT，是二分类 LOGIT 模型（别名为 CLOGIT），也适合于两个响应类以上的累积 LOGIT 模型（别名为 CUMLOGIT）。如果想用其他函数，可以通过 LINK=指定函数名。可以取的其他函数如下：CLOGLOG（互补双对数函数）、GLOGIT（广义 LOGIT 函数）和 PROBIT（概率单位模型）。

NOINT：不使用截距。

NOFIT：不使用模型拟合。

OFFSET=：指定偏移变量名。

SELECTION=：指定回归分析中选择的模型，默认为 NONE，简写为 N，为完全模型，根据指定的模型大小，找出最佳模型。其他 4 个模型可以根据分析需求选择。

- SELECTION=FORWARD：FORWARD 模型是向前回归模型，按照 SLE 规定的 P 值，根据 F 检验和 T 检验计算残差平方和作为引入变量到模型中的条件，如果第一个引入

模型的变量通过这些检验就保留此变量，然后依次引入第二个变量，并按照引入的变量检验看第二个引入变量是否符合，不符合就剔除，依次类推引入其他变量。

- SELECTION=BACKWARD：BACKWARD 是向后回归模型，与 FORWARD 正好相反。开始所有的变量都在模型中，按照 SLE 规定的 P 值从含有全部变量的模型开始，并对第一次引入模型的变量进行 F 检验和 T 检验，如果此变量不符合检验条件就剔除，其他变量依次类推。
- SELECTION=STEPWISE：STEPWISE 是逐步回归模型。按照 SLE 的标准，综合向前和向后回归模型的方法，根据引入变量进行的偏回归平方和进行检验，如果此变量经过条件检验比较显著就引入到模型中同时对已经引入模型中的变量再用偏回归平方和进行检验。如果有不显著变量就剔除，其他变量依次类推。
- SELECTION=SCORE：最优子集模型。

【注意】 SLECTTION=选择模型时截距是被强制保留在模型中的，如果不需要截距需要用选项“NOINT”指定，表示剔除截距。

（1）指定 MODEL 语句中模型建立的详细粒度

BEST=：对于指定 SCORE 选项的，可以通过此选项指定模型显示的数据。

DETAILS：指定每步详细的信息结果。

FAST：模型建立中使用 FAST 排除方法。

HIERARCHY=：指定模型每步的层次。

MAXSTEP=：对于模型选择方式选择的 STEPWISE（逐步引进变量方式）指定最大步数。

SEQUENTIA：对于模型中指定顺序筛选变量。

SLENTRY=：指定模型的显著性水平。

START=：指定第一个模型的变量数。

STOP=：指定最终模型的变量数。

STOPRES：对于变量添加或删除的残差卡方标准。

（2）指定模型拟合选择项

ABSFCONV=：指定绝对函数收敛性判别标准。

CONV=：指定相关函数收敛性判别标准。

GCONV=：指定相关剃度收敛性判别标准。

XCONV=：指定相关参数收敛性判别标准。

MAXITER=：指定模型中的迭代最大数。

SINGULAR=：指定模型中奇异性检验的容忍值。

TECHNIQUE=：指定模型中的最大极值的迭代算法。

（3）指定模型中的置信区间选择项

ALPHA=：指定 α 概率值，是相对于回归参数和优势比（Odds Ratio）置信区间为 100（1−α）%概率值而言的，α 的取值范围为[0,1]。默认 ALPHA=0.05。

（4）指定模型中的分类表选择项

CTABLE：此选项可以显示分类表。

PEVENT=：指定模型中的先验事件概率，分层抽样用这个选择项。

PPROB=：对于指定 CTABLE 分类表观测可以通过此选项指定临界概率值，取值范围为[0，1]。

（5）指定模型中的 ROC 曲线选择项

OUTROC=：对于二分类响应模型指定输出的 ROC 曲线数据集的名字。

ROCEPS=：指定似然性组标准概率，取值范围为[0，1]，默认值为 1E-4。

（6）指定模型中的回归诊断选择项

INFLUENCE：对于二分类响应模型中为鉴别有影响观测显示诊断尺度。

IPLOTS：对于模型中的每一个回归诊断统计生成索引散点图。

（7）指定模型中的详细显示选择项

CORRB：显示参数估计的相关矩阵。

COVB：显示参数估计的协方差矩阵。

【注意】如果用 CLASS 语句分类时，其必须放在 MODEL 语句前面；如果用 CONTRAST 语句时，其必须放在 MODEL 语句之后。

7.3.2 相关分析与回归分析应用

1．相关分析应用案例

相关分析在医学、经济领域以及产品检验领域经常用到，研究变量之间的关系以及关联强度。

【例 7.9】 研究遗传学身高，测量 20 名学生的身高与其父母的身高数据，如表 7-9 所示。

表 7-9 遗传学身高数据

	本 人		父 亲	母 亲
	性 别	身高/cm	身高/cm	身高/cm
1	男	164	160	155
2	女	156	160	157
3	男	168	160	165
4	女	160	175	156
5	女	162	163	155
6	男	187	181	156
7	女	162	172	160
8	男	167	175	158
9	女	160.5	175	162
10	女	160	167	162
11	女	158	172	155
12	女	164	169	163
13	女	165	172	160
14	男	174	175	162

（续）

	本　　人		父　　亲	母　　亲
15	女	166	172	165
16	女	158	170	159
17	男	162	158	147
18	男	175	167	163
19	男	170	164	157
20	女	161	176	160

```
*对外部数据处理;
%let   path= D:\jx\shengao;   /*定义外部文件路径*/
%let   type=.txt;
%let   fil= "&path&type";
libname   jx 'd:\jx';   /*定义逻辑库*/
data   jx.shengao;   /*数据集存储到指定逻辑库*/
       Infile   &fil   dlm='|'   dsd   missover;
            input                    haizi      :4.
                                     father    :4.
                                     Mother    :4.
                 ;
run;
/*调 CORR 过程分析*/
proc   corr   data=jx.shengao   nosimple;
      var   father   mother   haizi;
run;
```

程序执行后输出窗口显示如图 7-10 所示。

```
输出 - （无标题）
                    SAS 系统                2012年06月29
                 CORR PROCEDURE

         3  变量:    father    mother    haizi

              Pearson 相关系数, N = 20
            当 H0: Rho=0 时，Prob > |r|

                 father      mother       haizi

father          1.00000     0.27652     0.32328
                             0.2379      0.1644

mother          0.27652     1.00000     0.14568
                 0.2379                  0.5400

haizi           0.32328     0.14568     1.00000
                 0.1644      0.5400
```

图 7-10　遗传学身高分析

【程序解读】

1）PROC corr data=jx.shengao nosimple：调用 CORR 过程，nosimple 选项抑制简单统计。

2）var father mother haizi：分析这 3 个变量的相关性。

【分析结果解读】

分析结果显示第一部分显示简单统计，3 为变量个数，分别是 father、mother 和 haizi。

第二部分显示 Pearson 相关系数矩阵。看 haizi 这一行，第一行为相关系数，与 father 变量相关系数 r=0.32328，与 mother 相关系数 r=0.14568。第二行为 p 值，是当 H0:Rho=0 时，prob>|r|，即检验“相关系数 Rho 为 0”时的显著性概率。此处与 father 变量绝对不相关的概率 p=0.1644，与 mother 变量绝对不相关的概率 p=0.5400。

通过分析可以看出孩子的身高与父母的身高是正相关的。

2．REG回归分析应用案例

相关分析只是研究变量之间是否有关系。变量之间存在什么关系，实际上还需要进一步分析。

【例 7.10】 回归分析，对例 7.9 通过回归分析研究孩子身高与父母身高的关系。

【提示】 例 7.9 通过相关分析验证变量之间有关系，可以通过回归分析确定变量之间是什么关系。

```
 *对外部数据处理;
%let   path= D:\jx\shengao;   /*定义外部文件路径*/
%let   type=.txt;
%let   fil= "&path&type";
libname   jx 'd:\jx';   /*定义逻辑库*/
data   jx.shengao;   /*数据集存储到指定逻辑库*/
       Infile   &fil   dlm='|'   dsd   missover;
                 input                    haizi    :4.
                                          father   :4.
                                          mother   :4.
                  ;
run;
/*调 REG 过程分析*/
proc   reg   data=jx.shengao ;
   model   haizi=father   mother / noint   influence;
run;
quit;
```

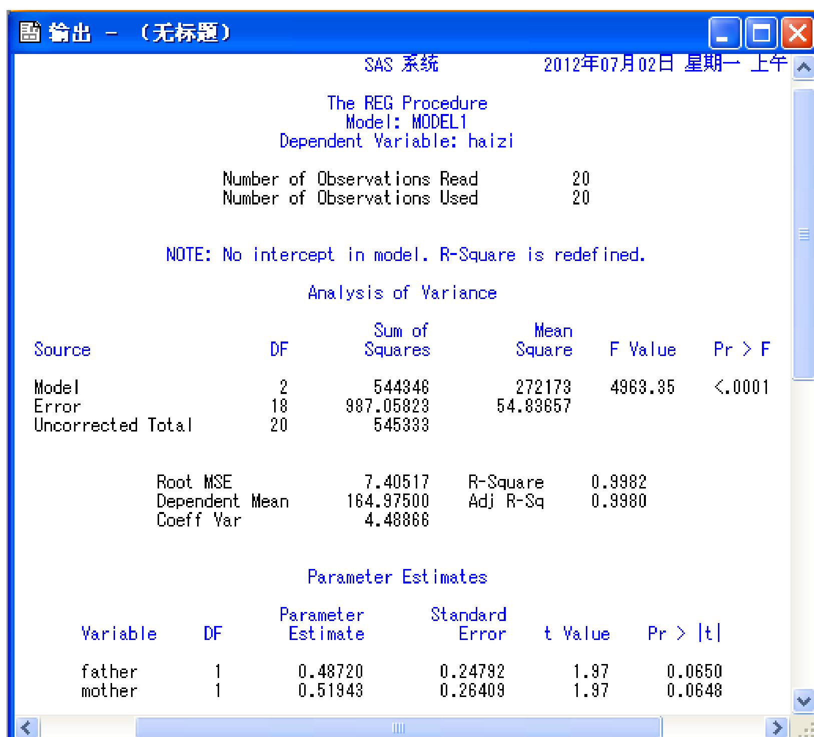
程序执行后输出窗口显示如图 7-11 所示。

【程序解读】

model haizi=father mother / noint influence：因变量为 haizi（孩子身高），回归变量表为 father（父亲身高）和 mother（母亲身高），noint 选项指定模型拟合时不包含截距项，influence 选项要求详细分析每个观测对估计和预测值的影响。

【分析结果解读】

根据回归分析确定孩子身高与父母身高程什么线性关系，图 7-11（一）输出回归的t检验及方差分析，其中F=4963.35，P<0.0001，R^2为 0.9982，说明直线回归方程非常显著。根据参数估计，通过原点的直线回归方程得到孩子身高为：haizi=0.48720father+0.51943mother，可以预测到未来孩子身高大约值=0.48720*父亲身高+0.51943*母亲身高。



输出 - （无标题）

SAS 系统　　2012年07月02日 星期一 上午

The REG Procedure
Model: MODEL1
Dependent Variable: haizi

Number of Observations Read　20
Number of Observations Used　20

NOTE: No intercept in model. R-Square is redefined.

Analysis of Variance

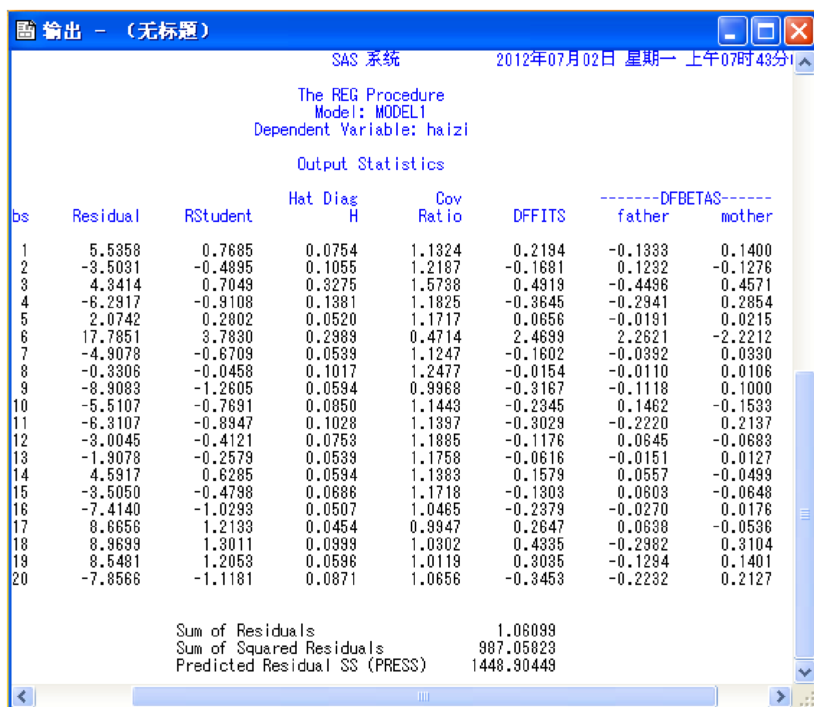
Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	2	544346	272173	4963.35	<.0001
Error	18	987.05823	54.83657		
Uncorrected Total	20	545333			

Root MSE	7.40517	R-Square	0.9982
Dependent Mean	164.97500	Adj R-Sq	0.9980
Coeff Var	4.48866		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
father	1	0.48720	0.24792	1.97	0.0650
mother	1	0.51943	0.26409	1.97	0.0648

REG 回归分析（一）



输出 - （无标题）

SAS 系统　　2012年07月02日 星期一 上午07时43分

The REG Procedure
Model: MODEL1
Dependent Variable: haizi

Output Statistics

bs	Residual	RStudent	Hat Diag H	Cov Ratio	DFFITS	-------DFBETAS------ father	mother
1	5.5358	0.7685	0.0754	1.1324	0.2194	-0.1333	0.1400
2	-3.5031	-0.4895	0.1055	1.2187	-0.1681	0.1232	-0.1276
3	4.3414	0.7049	0.3275	1.5738	0.4919	-0.4496	0.4571
4	-6.2917	-0.9108	0.1381	1.1825	-0.3645	-0.2941	0.2854
5	2.0742	0.2802	0.0520	1.1717	0.0656	-0.0191	0.0215
6	17.7851	3.7830	0.2989	0.4714	2.4699	2.2621	-2.2212
7	-4.9078	-0.6709	0.0539	1.1247	-0.1602	-0.0392	0.0330
8	-0.3306	-0.0458	0.1017	1.2477	-0.0154	-0.0110	0.0106
9	-8.9083	-1.2605	0.0594	0.9968	-0.3167	-0.1118	0.1000
10	-5.5107	-0.7691	0.0850	1.1443	-0.2345	0.1462	-0.1533
11	-6.3107	-0.8947	0.1028	1.1397	-0.3029	-0.2220	0.2137
12	-3.0045	-0.4121	0.0753	1.1885	-0.1176	0.0645	-0.0683
13	-1.9078	-0.2579	0.0539	1.1758	-0.0616	-0.0151	0.0127
14	4.5917	0.6285	0.0594	1.1383	0.1579	0.0557	-0.0499
15	-3.5050	-0.4798	0.0686	1.1718	-0.1303	0.0603	-0.0648
16	-7.4140	-1.0293	0.0507	1.0465	-0.2379	-0.0270	0.0176
17	8.6656	1.2133	0.0454	0.9947	0.2647	0.0638	-0.0536
18	8.9699	1.3011	0.0999	1.0302	0.4335	-0.2982	0.3104
19	8.5481	1.2053	0.0596	1.0119	0.3035	-0.1294	0.1401
20	-7.8566	-1.1181	0.0871	1.0656	-0.3453	-0.2232	0.2127

Sum of Residuals	1.06099
Sum of Squared Residuals	987.05823
Predicted Residual SS (PRESS)	1448.90449

REG 回归分析（二）

图 7-11

3．LOGISTIC回归分析应用案例

LOGISTIC 数据挖掘中经常用到此模型，这个模型是二事件模型。

【例 7.11】 请根据客户购买电子产品的数据分析哪类人员购买电子产品比较多，以便于营销人员有针对性的推广。购买结果说明如表 7-10 所示。客户学历类型说明如表 7-11 所示。

表 7-10　购买结果说明

购 买 标 识	购买标识说明
0	客户购买电子产品
1	客户没有购买电子产品

表 7-11　客户学历类型说明

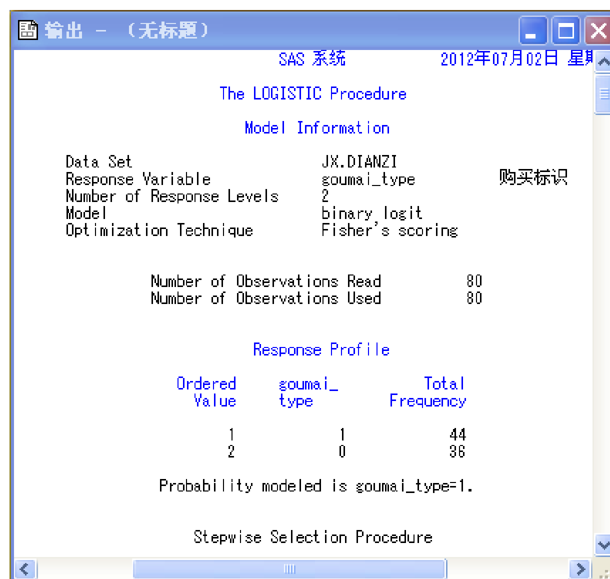
学历标识	学历标识说明
1	博士
2	硕士
3	本科
4	专科
5	高中
6	初中
7	小学

回归分析程序如下：

```
*对外部数据处理;
%let   path= D:\jx\dianzi;   /*定义外部文件路径*/
%let   type=.txt;
%let   fil= "&path&type";
libname   jx 'd:\jx';   /*定义逻辑库*/
data   jx.dianzi;   /*数据集存储到指定逻辑库*/
       Infile   &fil ;
/*文件记录长度超过 256 用 lrecl=指定长度*/
input   id   $ name   $   education   yincome   work_period   goumai_days   goumai_type;
    label   goumai_type='购买标识' ;
run;
proc   logistic   data=jx.dianzi   desc;/*逻辑回归分析，降序排列目标变量*/

       model   goumai_type=education   yincome   work_period /noint
                    selection=stepwise
                    sle=0.2
                    sls=0.1
                    details
                    stb;
     output   out=jx.goumai_analy;   /*指定输出分析结果到存储目录数据集*/
run;
```

程序执行后输出窗口显示如图 7-12～图 7-16 所示。



输出 - （无标题）

SAS 系统　　2012年07月02日 星

The LOGISTIC Procedure

Model Information

Data Set	JX.DIANZI	
Response Variable	goumai_type	购买标识
Number of Response Levels	2	
Model	binary logit	
Optimization Technique	Fisher's scoring	

Number of Observations Read	80
Number of Observations Used	80

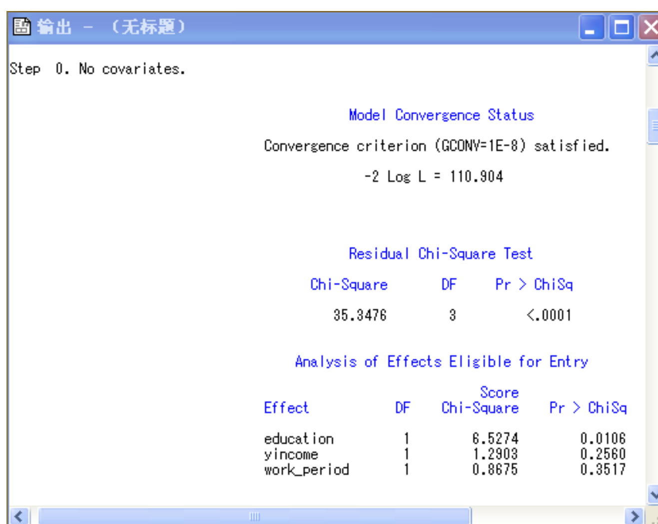
Response Profile

Ordered Value	goumai_type	Total Frequency
1	1	44
2	0	36

Probability modeled is goumai_type=1.

Stepwise Selection Procedure

图 7-12　LOGISTIC 逻辑回归-模型信息



输出 - （无标题）

Step 0. No covariates.

Model Convergence Status

Convergence criterion (GCONV=1E-8) satisfied.

-2 Log L = 110.904

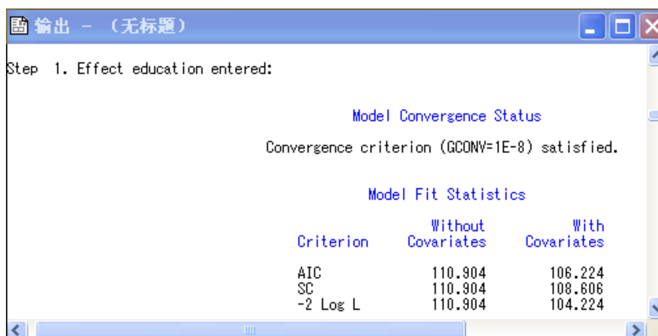
Residual Chi-Square Test

Chi-Square	DF	Pr > ChiSq
35.3476	3	<.0001

Analysis of Effects Eligible for Entry

Effect	DF	Score Chi-Square	Pr > ChiSq
education	1	6.5274	0.0106
yincome	1	1.2903	0.2560
work_period	1	0.8675	0.3517

图 7-13　LOGISTIC 逻辑回归-逐步选择 1



输出 - （无标题）

Step 1. Effect education entered:

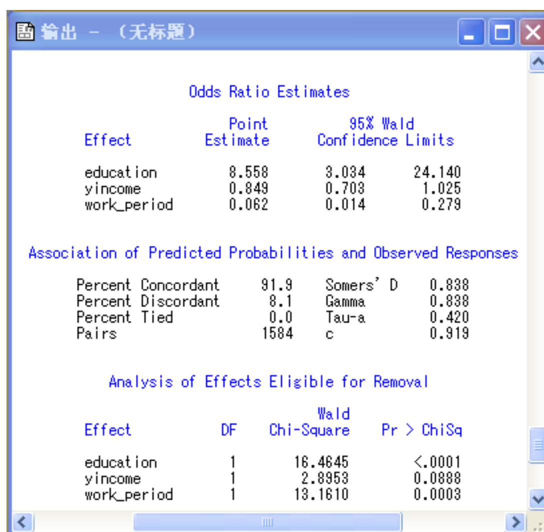
Model Convergence Status

Convergence criterion (GCONV=1E-8) satisfied.

Model Fit Statistics

Criterion	Without Covariates	With Covariates
AIC	110.904	106.224
SC	110.904	108.606
-2 Log L	110.904	104.224

图 7-14　LOGISTIC 逻辑回归-逐步法选择自变量



输出 －（无标题）

Odds Ratio Estimates

Effect	Point Estimate	95% Wald Confidence Limits	
education	8.558	3.034	24.140
yincome	0.849	0.703	1.025
work_period	0.062	0.014	0.279

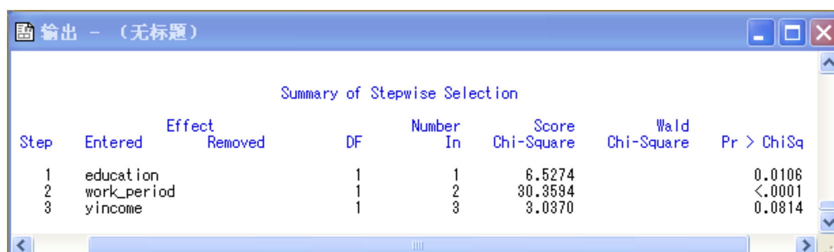
Association of Predicted Probabilities and Observed Responses

Percent Concordant	91.9	Somers' D	0.838
Percent Discordant	8.1	Gamma	0.838
Percent Tied	0.0	Tau-a	0.420
Pairs	1584	c	0.919

Analysis of Effects Eligible for Removal

Effect	DF	Wald Chi-Square	Pr > ChiSq
education	1	16.4645	<.0001
yincome	1	2.8953	0.0888
work_period	1	13.1610	0.0003

图 7-15　LOGISTIC 逻辑回归-OR 值估计



输出 －（无标题）

Summary of Stepwise Selection

Step	Effect Entered	Effect Removed	DF	Number In	Score Chi-Square	Wald Chi-Square	Pr > ChiSq
1	education		1	1	6.5274		0.0106
2	work_period		1	2	30.3594		<.0001
3	yincome		1	3	3.0370		0.0814

图 7-16　LOGISTIC 逻辑回归-逐步法选择自变量过程总结

【程序解读】

1）PROC logistic data=jx.dianzi desc：调用 LOGISTIC 逻辑回归过程，通过 data=语句指定进行 LOGISTIC 分析的数据集。

2）desc：此选项表示对数据集目标变量按照降序排序输出分析。

3）Model：指定建模的因变量=自变量。此程序 goumai_type 是因变量；自变量是 education、yincome 和 work_period 三个变量。

4）noint：noint 选项表示不要截矩。

5）Selection=stepwise：表示回归选择的模型，此程序选择的是逐步回归分析 stepwise。

6）sle=：筛选变量进入模型的阈值。

7）sls=：筛选变量保留在模型的阈值。

8）details：此选项指定把分析结果的详细信息输出。

9）std：此选项指定输出标准化数据模型选项。

10）output out=：指定把输出分析结果保存到指定数据集存储。

【分析结果解读】

1）图 7-12 所显示模型的信息如下。

Data Set：数据集信息行显示数据集为 JX.DIANZHI。

Response Variable：响应变量行，因变量为 goumai_type。

Number of Response Levels：响应变量分类数，也就是因变量可以取的值，此处为二分类，显示为 2。

Model：模型行指明模型 LINK=映射函数，此处为二分类 logit 函数。

Optimization Technique：优化技术行指明选择的优化技术为 Fisher’s scoring 方法。

Number of Observations Reed：此行指明分析选择的观测记录数为 80 条。

Response Profile：显示响应变量的信息，Ordered Value 指明响应变量的排列顺序；同时可以看出 goumai_type（购买标识）变量中 goumai_type=1 的购买客户为 44 位客户，goumai_type=0 表示没有购买的客户为 36 位客户；Probability model is default_type=1 表示计算用的分析响应变量的概率模型值为 1 事件的发生概率。

2）图 7-13 显示逐步回归分析过程信息如下。

Step 0. No covariates：表示逐步回归分析开始步信息，此处为第零步。

Model Convergence Status：模型收敛状态信息，此处对选择参数判断是否收敛的标准显示信息为 Convergence criterion（GCONV=1E-8）satisfied.，选择标准为 1E-8。该信息是用来对模型分析的，和变量无关，模型拟合要达到收敛标准。

Residual Chi-Square Test：此项下面显示残差信息，Chi-Square 显示残差卡方信息为 35 .3476；概率值 $p<0.0001$。

Analysis of Effects Eligible for Entry：此信息项对应的是引入自变量有效性信息分析输出的显示结果。

3）如图 7-14 所示，模型中含 3 个自变量的逐步分析，为便于理解，选择第一个变量 education 进行分析讲解，其他变量进行逐步分析的过程，和第一个变量 education 的引入方式和参数选择项一样。显示引进变量 education 的信息，对自变量的分析统计信息如下。

Model Fit Statistics：模型拟合统计量，在此项信息下面显示模型拟合指标信息。

Criterion：此列为模型拟合似然函数评价指标标准项。

AIC：为 Akaike Informatiom Criterion 的简写，是 Akaike 信息评价标准，公式表示为 AIC=–2LOGL+2(N+R)，–2LOGL 的取值范围为[0，∞]，值越小说明拟合越好；N 表示模型中自变量的数目；R 为响应变量（因变量）类别总数减 1，对于逻辑回归 R=2-1=1。模型中参数数量越大时，似然函数值也就越大，–2LOGL 变小。2（N+R）可以抵消参数数量产生的影响，AIC 值较小时表示模型拟合较好。

SC：为 Schwarz 标准，是根据自变量数目和观测数量对–2LOGL 值进行的一种调整，公式表示为 SC=–2LOGL+2（N+R）*ln(n)，此指标只能用于比较对同一数据所设的不同模型，其他条件相同时 SC 越小说明模型拟合越好。

–2LOGL：该计算公式类似于线性回归的总平方和。

Without Covariates：此列显示无变异系数时 3 个评价指标对应的值。

With Covariates：此列显示有变异系数时 3 个评价指对应的值。

4）如图 7-15 所示，回归分析最后显示对 3 个自变量的优势比估计，Odds Ratio Estimates 项显示了对 educaion、yincome 和 work_period 三个自变量的 Point Estimate（点估计）和 95%Wald Cofidence Limits（95%可信区间估计）信息。显示自变量 education（教育）占比较高，可以看出受教育水平越高的客户购买电子产品的可能性越大，学历越低的客户购买的电子产品越少；Association of Predicted Probabilites and Observed Response 项为预测概率和观测响应之间

的关联统计信息，此处重点关注 Pairs 项。c 为统计量，反映了实际观测值和模型预测的事件概率的关联强度，是一个很重要的指标。此实验 c 的对应值为 0.919，可以看出预测事件概率和实际观测数据之间的关联强度非常高。

5）如图 7-16 所示，最后输出回归分析过程汇总信息，显示选入变量的 P 值均小于 0.1，3 个自变量都保留在模型中。

结论：可以看出教育、工作年限和年收入这三个自变量参数可以用来判断哪些客户经常购买电子产品，对学历低、工作年限时间不长、年收入低的客户建议不做推广销售。

7.4　因子分析基础

因子分析是指研究从变量群中提取共性因子的统计技术。通过因子分析分析一种用少数的彼此不相关的综合指标来表达多个观测变量的多元统计方法。因子分析最早由英国心理学家 C.E.斯皮尔曼提出。他发现学生的各科成绩之间存在着一定的相关性，一科成绩好的学生，往往其他各科成绩也比较好，从而推想是否存在某些潜在的共性因子，或者某些一般智力条件影响着学生的学习成绩。因子分析可在许多变量中找出隐藏的具有代表性的因子。将相同本质的变量归入一个因子，可以减少变量的数目，还可以检验变量间的假设关系。

7.4.1　因子分析概述

因子分析在心理学、社会学、医学和经济学等学科中有着广泛的应用。因子分析的主要目的是用来描述隐藏在一组测量到的变量中的一些更基本的，但又无法直接测量到的隐性变量。例如，如果要测量通货膨胀的原因，是内部行政手段干预调控的直接原因还是其他原因。这里，通货膨胀与经济其他指标是无法直接用一个测度（比如一个问题）来测量的，它们必须用一组测度方法来测量，然后把测量结果结合起来，才能更准确地来把握。可以直接测量的可能只是它所反映的一个表征（manifest），或者是它的一部分。在这里，表征与部分是两个不同的概念。表征是由这个隐性变量直接决定的。隐性变量是因，而表征是果。例如，通货膨胀是货币发行过度的一个主要决定因素。

从显性的变量中得到因子，需要借助因子分析的方法，一类是探索性因子分析，另一类是验证性因子分析。探索性因子分析不事先假定因子与测度项之间的关系，而让数据“自己说话”。主成分分析是其中的典型方法。验证性因子分析假定因子与测度项的关系是部分知道的，即哪个测度项对应于哪个因子，虽然尚且不知道具体的系数。

1．探索性因子分析

因子分析的方法有重心法、影像分析法、最大似然解、最小平方法、拉奥典型抽因法等。这些方法本质上大都属近似方法，是以相关系数矩阵为基础的，所不同的是相关系数矩阵对角线上的值采用不同的共同性估值。在社会学研究中，因子分析常采用以主成分分析为基础的反覆法。主成分分析的目的与因子分析不同，它不是抽取变量群中的共性因子，而是将变量进行线性组合，成为互为正交的新变量，以确保新变量具有最大的方差。在求解中，正如因子分析一样，要用到相关系数矩阵或协方差矩阵。为了确定因子的实际内容，还须进一步旋转因子，使每一个变量尽量只负荷于一个因子之上，这就是简单的结构准则。常用的旋转有直角旋转法和斜角旋转法。作直角旋转时，各因素仍保持相对独立。在作斜角旋转时，允

许因素间存在一定关系。上述从变量群中提取共性因子的方法，又称 R 型因子分析和 R 型主要成分分析。但如果研究个案群的共性因子，则称 Q 型因子分析和 Q 型主成分分析。因子分析是社会研究的一种有力工具，但不能肯定地说一项研究中含有几个因子，当研究中选择的变量变化时，因子的数量也要变化。此外对每个因子实际含义的解释也不是绝对的。

2．验证性因子分析

验证性因子分析（Confirmatory Factor Analysis）的强项在于它允许研究者明确描述一个理论模型中的细节。因为测量误差的存在，研究者需要使用多个测度项。当使用多个测度项之后，就有测度项的“质量”问题，即有效性检验。而有效性检验就是要看一个测度项是否与其所设计的因子有显著的载荷，并与其不相干的因子没有显著的载荷。可能进一步检验一个测度项工具中是否存在单一方法偏差，一些测度项之间是否存在“子因子”。这些测试都要求研究者明确描述测度项、因子、残差之间的关系。对这种关系的描述又叫测度模型（Measurement Model）。对测度模型的质量检验是假设检验之前的必要步骤。验证性因子分析往往用极大似然估计法求解。它往往与结构方程的方法连用。

FACTOR 过程是 SAS 系统实现因子分析的过程。

语法格式：
```
PROC   FACTOR   < 选项>;
       PRIORS   数值 1，数值 2…;
       VAR  变量表;
       PARTIAL  变量表;
       FREQ   变量表 ;
       BY  变量表;
       WEIGHT   变量  </ 选项 >;
RUN;
```

【语法解读】

1）可以取的选项如下。

DATA＝SAS 数据集：给出输入数据集的名称。

OUTSAS 数据集：创建一个数据集，它包括来自 DATA＝的数据集中的全部数据，还包括被称为 Factor1、Factor2 等变量的因子得分估计。

OUTSTAT＝SAS 数据集：规定一个包含大部分分析结果的输出数据。

COV：要求用协方差阵替代相关系数阵作因子分析。

NFACTORS=n：规定被保留的因子个数。

ROTATE=name：规定旋转方法。

ROTATE=E：规定正交的均方最大旋转。

ROTATE=no：规定不旋转。

ROTATE=v：规定正交的方差最大旋转。

ALL：打印除图形之外的所有可选择的输出。

SCORE：打印因子得分系数。

2）PRIORS　数值 1，数值 2…：为每个变量指定一个介于 0.0 与 1.0 之间的先验公因子方差估计值。给出的数值与 VAR 语句中变量的顺序相对应，数值的个数与变量的数目要相等。

3）PARTIAL　变量表：指定基于偏相关或协方差矩阵进行分析的相应变量。

7.4.2 因子分析应用

因子分析主要应用在社会学、经济学、心理学和医学等领域中。

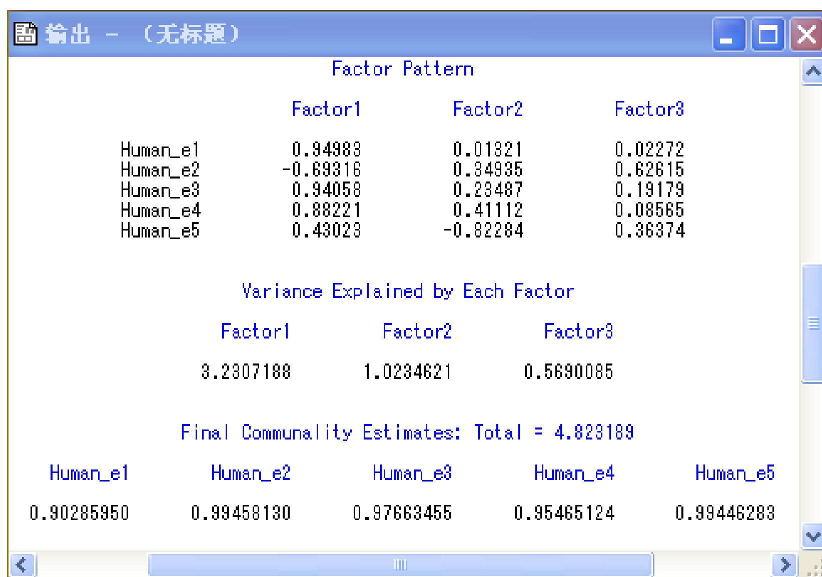
【例 7.12】为了全面提高人们的生活质量，满足人们日益增长的物质和文化的合理需求。在可持续发展消费的统一理念下，增加社会财富，创造更多的物质文明和精神文明，保持人类的健康延续和生生不息，维系人类与自然的平衡，提供整体物资文明和精神文明建设，国家采用“人文与经济（Humanity and Economy）发展综合系数”指标对于国民生活质量进行测度。人文发展综合系数指标组合，即人的健康状况、教育程度、人均国民收入，通过这 3 类指标组合的整体表达内涵，去衡量一个国家或地区的社会发展总体状况以及国民生活质量的总水平。指标体系如表 7-12 所示。

表 7-12 人文与经济发展指标体系

指 标 标 识	指标标识说明
area	地区
Human_e1	人均 GDP
Human_e2	人口总数
Human_e3	教育程度（1-13）
Human_e4	房价/m^2
Human_e5	预期寿命

```
data    Human_e;
   title "国民生活质量人文与经济指标分析";
   input    area    $    Human_e1-Human_e5 ;
cards;
上海  10827 23019148 11.6 15258 74
天津  10399 12938224 10.3 7979 76
北京  10377 19612368 12.6 18300 75
江苏  7779 78659903 8.2 5817 76
浙江  7524 54426891 8.3 9279 70
内蒙古  6978 24706321 7.5 3518 73
广东  6440 104303132 8.9 7482 72
辽宁  6232 43746323 7.3 4448 75
山东  6040 95793065 8.1 3919 73
福建  5900 36894216 8.3 6250 74
;
run;
/*因子分析过程调用*/
proc    factor    data=Human_e nfactors=3    score    out=analy_hum_e;
run;
```

程序执行后输出窗口显示如图 7-17 所示。



输出 - （无标题）

Factor Pattern

	Factor1	Factor2	Factor3
Human_e1	0.94983	0.01321	0.02272
Human_e2	-0.69316	0.34935	0.62615
Human_e3	0.94058	0.23487	0.19179
Human_e4	0.88221	0.41112	0.08565
Human_e5	0.43023	-0.82284	0.36374

Variance Explained by Each Factor

Factor1	Factor2	Factor3
3.2307188	1.0234621	0.5690085

Final Communality Estimates: Total = 4.823189

Human_e1	Human_e2	Human_e3	Human_e4	Human_e5
0.90285950	0.99458130	0.97663455	0.95465124	0.99446283

因子分析（一）



输出 - （无标题）

国民生活质量人文与经济指标分析

2012年07月04日

The FACTOR Procedure
Initial Factor Method: Principal Components

Scoring Coefficients Estimated by Regression

Squared Multiple Correlations of the Variables with Each Factor

Factor1	Factor2	Factor3
1.0000000	1.0000000	1.0000000

Standardized Scoring Coefficients

	Factor1	Factor2	Factor3
Human_e1	0.29399823	0.01290983	0.0399306
Human_e2	-0.2145533	0.34133891	1.10042421
Human_e3	0.29113597	0.22948656	0.33705544
Human_e4	0.27306932	0.40169676	0.15052154
Human_e5	0.13316732	-0.8039729	0.63925884

因子分析（二）

图 7-17

【程序解读】

PROC factor data=Human_e nfactors=3 score out=analy_hum_e：nfactors=3 指定将要提取因子分最大数目为 3，score 指定显示特征值碎石图。

【分析结果解读】

根据因子分析矩阵可以看出 Human_e1（人均 GDP）在因子 factor1 占比 0.29399823（最高），Human_e4（房价）在因子 factor2 为 0.40169676（最高），Human_e5（预期寿命）在因子 factor3 为 0.63925884（最高），从而推断出人均 GDP、房价和预期寿命是影响国民人文与经济满意度的主要原因，需要加强改善这些指标，以便于提高国民生活质量。

7.5 生存分析基础

生存分析（Survival Analysis）主要应用在医学、生物学、保险学、可靠性工程学、社会学、经济学等方面。生存分析是对生存资料的分析，根据试验或调查得到的数据对生物或人的生存时间进行分析和推断，研究生存时间和结局与众多影响因素间的关系及其程度大小的方法，也称生存率分析或存活率分析。

7.5.1 生存分析概述

生存分析是研究生存现象和响应时间数据及其统计规律的一门学科，其目的是描述生存时间的分布以及它与其他变量的关系。生存分析的响应变量是生存时间，与其有关的自变量是与生存时间有关的其他变量，其中自变量可以是离散的（如性别、高矮、民族等），也可以是连续变量（如温度、身高和年龄等）。

1．生存分析方法

（1）非参数法

非参数检验是检验分组变量各水平所对应的生存取消是否一致。常用的方法有乘积极限法、寿命表法、对数秩检验和似然比检验等。

（2）参数法

参数法是指已知生存时间服从特定的参数模型。常用方法有指数分布法、威布尔分布（Weibull Distribution）法、对数正态回归分析法以及对数 LOGISTIC 回归分析法等。

（3）半参数法

半参数法是在特定的假设条件下，创建生存时间随多个风险因素变化的回归方程。常用方法如 Cox 模型分析方法。

2．生存分析描述语

（1）生存时间

生存时间指观察体的存活时间，通常称为失效时间。必须指定生存时间，只能指定一个变量作为生存时间。生存时间有两种类型：完全数据时间（指从研究对象事件的起点到事件发生时所经历的时间）和截尾数据时间（对研究对象由于到研究时间结束时事件尚未发生等情况）。

（2）终检变量

终检变量值是指生存时间是否删失，为数值类型的非缺失值。

（3）层变量

层变量是生存分析中用来确定层水平的变量。

（4）检验协变量

检验协变量是生存分析中用来检验与失效时间相关的数值变量。

（5）ID 变量

ID 变量是生存分析中用来做观测的标签。

（6）生存概率

生存概率是指生存分析中研究对象开始时间到结束时间仍然存活的可能性大小，用 P 表示。

3. SAS系统常用生存分析过程

（1）lifetest 过程

lifetest 过程属于非参数法生存分析过程，主要用来分析有右终检值的数据，计算响应变量与其他变量相关联的秩次检验。这类数据分析时一般需要估计生存时间分析，用到生存分布函数 SDF（生存函数）。

```
语法格式：PROC lifetest < 选项 > ;
                  time 时间变量 < *终检变量表 > ;
                  by 变量表;
                  freq 变量;
                  id 变量表;
                  strata 变量 < 表 > < ... 变量 < (表) > > ;
                  survival 选项;
                  test 变量表;
          RUN;
```

【语法解读】

1）lifetest < 选项 >：lifetest 过程常用选项如下。

data=：指定生存分析的数据集。

Method=PL|LT|act：设置用以计算生存函数估计值的方法。此选项可设置的值及其含义如下：“pl”（或“km”）为乘积极限法（或 Kaplan-Meier 法）；“act”（或“life”或“lt”）为寿命表法。默认设置为“method=pl”。

Intervals=：寿命表指定用以计算寿命表的区间端点。区间端点必须为非负数，无论指定的区间端点中有无 0，寿命表的计算总是以 0 为第一个区间的起点，而每一个区间只具有下端点而无上端点（即为半开半闭区间）。

Width=：寿命表法计算生存函数估计值时，指定寿命表的区间宽度。如果指定了“intervals=”选项，此选项失效。

Plots=：生存函数估计值或删失值绘制图形。其完整设置形式应为“PLOTS= (type <(NAME=name)> <, ..., type <(NAME=name)> >)”，“type”代表绘图的类型，“NAME=name”表示将所绘图形以指定的名称存储为 SAS 目录的条目（entry）。其中，“NAME”为关键字（可为小写），“name”为用户指定的条目名称。关于绘图类型详见“plots=”选项的绘图类型。

alpha=：取值范围为 0.0001～0.9999 的小数，指定生存时间四分位数间距可信区间的置信水平，默认设置为“alpha=0.05”。

outtest=：指定生存分析的输出数据集。

Missing：此选项允许数值变量的缺项值或字符变量的空格值作为有效的分层水平。

2）time 时间变量 < *终检变量表 > ：time 语句指定失效时间变量，星号后为一右终检变量。

3）strata 变量 < 表 > < ... 变量 < (表) > > ：strata 语句的作用是指定用以分层的变量，如果要将 strata 变量的缺失值也作为一个合法的水平用以分层，可在 strata 语句中设置“missing”选项。strata 语句中可指定多个分层变量，分层情况将由各分层变量水平的组合来决定。对于数值型分层变量，若为其设置分层区间端点列表，该变量的水平与形成的区间一一对应，设置方式与 proc lifetest 语句的“intervals=”选项中类似；对于数值型分层变量，若

为其设置分层区间端点列表，该变量的水平与形成的区间一一对应。所划分的区间均为半开半闭区间，第一个区间总是以$-\infty$为下界，最后一个区间总是以∞为上界。

4）test 变量表：test 语句指定需要与生存时间进行关联性分析的协变量，语句中的变量必须为数值型。对于 test 变量，lifetest 过程将通过两类秩检验统计量来检验它们与生存时间的关联性，每一个 test 变量的单变量检验的统计量也将被给出，一个有关协变量联合效应的统计量列表也将被给出，此统计量列表的顺序和各协变量对联合效应贡献的大小顺序相一致。

（2）lifereg 过程

lifereg 过程属于参数法过程，又称为生存回归过程，用参数模型拟合可能有终检值的一组数据。

```
语法格式：PROC lifereg < 选项 > ;
              by 变量表;
              class 变量表;
              <标号：> model 响应变量=<效应></选项>;
              Output <out=数据集> 关键字=名< …关键字=名> <选项>;
          RUN;
```

【语法解读】

1）lifereg < 选项 >：常用选择项如下。

data=数据集名：指定生存分析的数据集。

Outtest=数据集名：指定输出数据集。

2）class 变量表：指定分类变量。注意，此语句必须在 model 语句之前出现。

3）model 响应变量=<效应></选项>：创建响应变量与自变量之间的关系模型。

4）output：指定将生存分析模型统计量输出到一个新的 SAS 数据集中。

（3）phreg 过程

phreg 过程属于半参数法生存分析过程，针对生存数据执行基于 Cox 比例风险模型（Cox Proportional Hazards Model）的回归分析，可以检验有关回归参数的线性假设，针对配对病例对照研究执行 LOGISTIC 回归分析过程，创建包含有关统计量的输出数据集等。

```
语法格式：PROC   phreg < 选项 >;
             model response < *censor(list) > = variables < /options > ;
             strata variable < (list) > < ...variable < (list) > > < /option > ;
             < label: > test equation1 < ,..., equationk > < /option > ;
             freq variable ;
             weight variable < /option >;
             id variables ;
             output < out=sas-data-set > < keyword=name... keyword=name > < /options > ;
             baseline < out=sas-data-set > < covariates=sas-data-set >
             < keyword=name... keyword=name > < /options > ;
             by variables ;
          RUN;
```

【语法解读】

1）phreg < 选项 >：常用选项如下。

data=数据集名：指定生存分析的数据集名。

covout：将各参数估计值的协方差矩阵输出到“outest=”选项所指定的输出数据集中，此选项只在设置了“outest=”选项的情况下才有效。

noprint：不打印输出分析结果到输出窗口。

nosummary：禁止结果中对删失值和非删失值观测频数的显示。

outest=：指定输出数据集，用以存储回归系数估计值等若干统计量。如果设置了“covout”选项，此输出数据集中将包含各参数的协方差矩阵。

simple：对于 model 语句中指定的自变量，输出结果中仅显示有关的简单统计量，如均数、标准差、最小值以及最大值等。

2）model 语句：model 语句指定作为失效时间的变量、可选的删失值状态变量以及自变量等。

model 语句可设置为两种不同的方式：

```
model response < *censor ( list ) > = variables < /options > ;
  model (t1, t2) < *censor(list) > = variables < /options > ;
```

第一种格式适用于仅有一个应变量的情况，第二种格式适用于计数过程输入方式的两个应变量的情况。model 语句的第一种类型中，等号前的“response”项代表作为失效时间的变量，如果包含删失值，则需指定“censor”项，以表示删失值状态；model 语句的第二种类型中，表示失效时间的变量为两个，构成一个半开半闭区间，表示观察对象处于危险状态的时间区间。phreg 过程要求删失值状态变量和自变量必须为数值型变量，失效时间变量不能取负数值。如果失效时间的取值为负数，相应的观测将被剔除。

7.5.2 生存分析应用

生存分析实际的应用比较广泛，为便于理解下面通过 3 个实际应用案例进行详细讲解。

【例 7.13】 对某类夏季服装生存期分析，6 月份销售情况数据如表 7-13 所示。

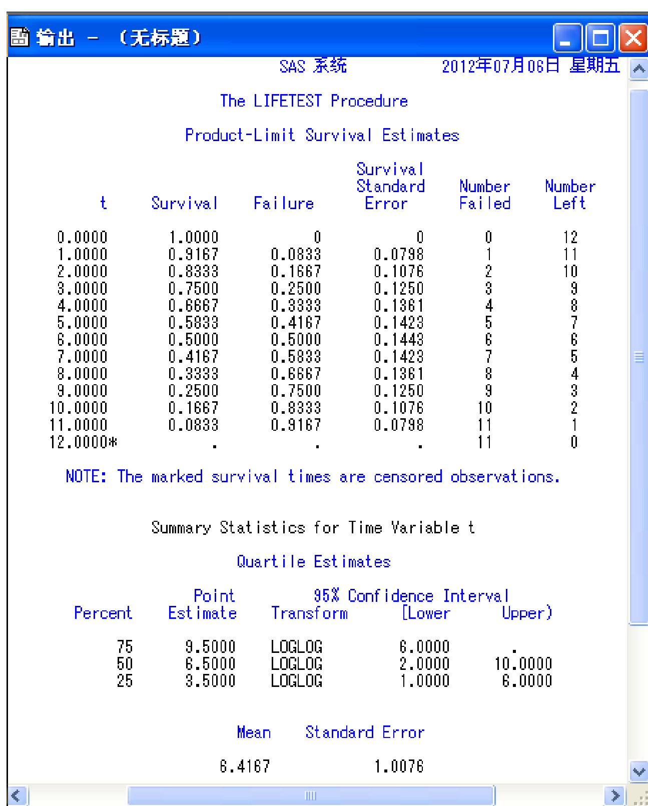
表 7-13 夏季服装销售情况

日 期	销 售 量	天 数	日 期	销 售 量	天 数
1	30	1	7	67	7
2	37	2	8	70	8
3	42	3	9	32	9
4	46	4	10	23	10
5	48	5	11	10	11
6	52	6	12	0	12

```
 data    summer_cloth;
 input    t    sale    days;
cards;
1       30     1
2       37     2
```

```
3    42    3
4    46    4
5    48    5
6    52    6
7    67    7
8    70    8
9    32    9
10   23    10
11   10    11
12   0     12
;
run;
proc   lifetest   data=summer_cloth   method=KM   plots=(s);
/*method=PL/KM/LT/LIFE/ACT 指定生存率估计方法*/
   time   t*sale(0);
/*必须语句，设置生存时间变量和生存结局变量，括号内为删失值*/
run;
```

程序执行后输出窗口显示如图 7-18～图 7-20 所示。



输出 - （无标题）

SAS 系统　　2012年07月06日 星期五

The LIFETEST Procedure

Product-Limit Survival Estimates

t	Survival	Failure	Survival Standard Error	Number Failed	Number Left
0.0000	1.0000	0	0	0	12
1.0000	0.9167	0.0833	0.0798	1	11
2.0000	0.8333	0.1667	0.1076	2	10
3.0000	0.7500	0.2500	0.1250	3	9
4.0000	0.6667	0.3333	0.1361	4	8
5.0000	0.5833	0.4167	0.1423	5	7
6.0000	0.5000	0.5000	0.1443	6	6
7.0000	0.4167	0.5833	0.1423	7	5
8.0000	0.3333	0.6667	0.1361	8	4
9.0000	0.2500	0.7500	0.1250	9	3
10.0000	0.1667	0.8333	0.1076	10	2
11.0000	0.0833	0.9167	0.0798	11	1
12.0000*	.	.	.	11	0

NOTE: The marked survival times are censored observations.

Summary Statistics for Time Variable t

Quartile Estimates

Percent	Point Estimate	Transform	95% Confidence Interval [Lower	Upper)
75	9.5000	LOGLOG	6.0000	.
50	6.5000	LOGLOG	2.0000	10.0000
25	3.5000	LOGLOG	1.0000	6.0000

Mean	Standard Error
6.4167	1.0076

图 7-18　乘积极限生存估计-寿命表

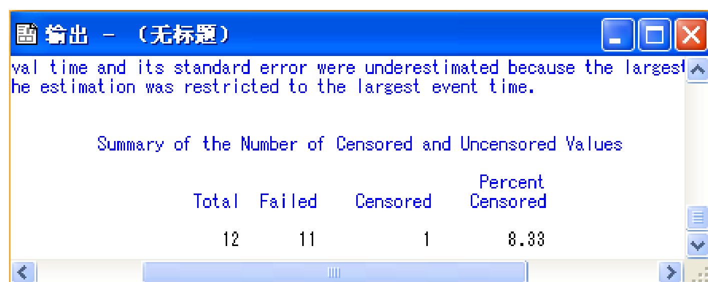


图 7-19　终检值和非终检值概述

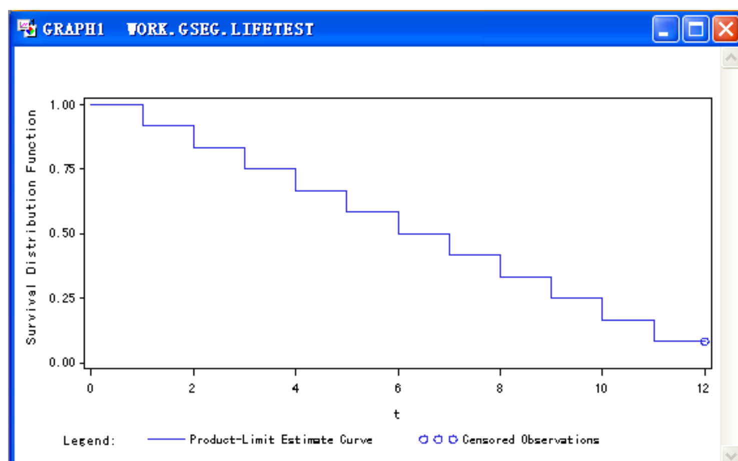


图 7-20　生成分析曲线图

【程序解读】

1）PROC lifetest data=summer_cloth method=KM plots=(s)：method=KM 指定生存率估计方法。

2）time t*sale(0)：必须语句，设置生存时间变量和生存结局变量，括号内为删失值。

【分析结果解读】

图 7-18 给出不同日数的生存率（Survival）、死亡率（Fauluer）、生存率的标准误（Survival Standard Error）和死亡数（Number Failed）。

图 7-19 所示服装退出市场 11，终检 1 件服装占总服装数 12 的 8.33%。

图 7-20 所示折线为乘积限估计曲线，“o”表示终检观测。

【例 7.14】 对例 7.13 通过 phreg 过程分析。

```
data   summer_cloth;
 input   t   sale   days;
cards;
1     30     1
2     37     2
3     42     3
4     46     4
5     48     5
6     52     6
7     67     7
```

```
8     70    8
9     32    9
10    23    10
11    10    11
12    0     12
;
run;
proc   phreg   data=summer_cloth ;
  model     t*days(12)=sale ;
run;
```

程序执行后输出窗口显示如图 7-21～图 7-23 所示。

```
输出 - (无标题)
                    SAS 系统              2012年07月09日

                 The PHREG Procedure

                  Model Information

      Data Set                    WORK.SUMMER_CLOTH
      Dependent Variable          t
      Censoring Variable          days
      Censoring Value(s)          12
      Ties Handling               BRESLOW

        Number of Observations Read          12
        Number of Observations Used          12

  Summary of the Number of Event and Censored Values

                                           Percent
        Total        Event     Censored    Censored

           12           11            1        8.33
```

图 7-21　比例风险分析（一）

```
输出 - (无标题)
                 Convergence Status

  Convergence criterion (GCONV=1E-8) satisfied.

                Model Fit Statistics

                      Without           With
   Criterion       Covariates     Covariates

   -2 LOG L            39.974         35.821
   AIC                 39.974         37.821
   SBC                 39.974         38.219
```

图 7-22　比例风险分析（二）

```
输出 - (无标题)
                Testing Global Null Hypothesis: BETA=0

         Test                  Chi-Square       DF      Pr > ChiSq

         Likelihood Ratio          4.1529        1          0.0416
         Score                     3.9867        1          0.0459
         Wald                      3.5688        1          0.0589

               Analysis of Maximum Likelihood Estimates

                        Parameter      Standard                                 Hazard
Parameter      DF        Estimate         Error    Chi-Square    Pr > ChiSq      Ratio

sale            1         0.03219       0.01704        3.5688        0.0589      1.033
```

图 7-23　比例风险分析（三）

【程序解读】

model t*days(12)=sale：t 为日期，days(12)为终检变量，终检值为 12，sale 为销售量。

【分析结果解读】

图 7-21 给出输出模型信息：数据集为 summer_cloth、应变量以及自变量的名称，终检变量取值为 12，对数据处理用 BRESLOW 方法，输出死亡数、终检数占总百分数为 8.33%。

图 7-22 所示为输出收敛状态，收敛标准为 GCONV=1E-8 的模型符合统计量、有变异和无变异的 3 种收敛值。

图 7-23 所示为输出整体无效假设检验：BETA=0，P 值都小于 0.0589，说明模型有统计意义。

7.6 聚类分析基础

聚类分析是研究事物分类的一种统计方法，是指将物理或抽象对象的集合分组成为由类似的对象组成的多个类的分析过程。它是一种重要的人类行为。聚类分析的目标就是在相似的基础上收集数据来分类。聚类源于很多领域，包括数学、计算机科学、统计学、生物学和经济学。在不同的应用领域，很多聚类技术都得到了发展，这些技术方法被用做描述数据，衡量不同数据源间的相似性，以及把数据源分类到不同的簇中，根据数据特征进行的分类研究。

7.6.1 聚类分析概述

聚类与分类的不同在于聚类所要求划分的类是事先未知的。聚类是将数据分类到不同的类或者簇的一个过程，同一个簇中的对象有很大的相似性，而不同簇间的对象有很大的相异性。

从统计学的观点来看，聚类分析是通过数据建模简化数据的一种方法。传统的统计聚类分析方法包括系统聚类法、分解法、加入法、动态聚类法、有序样品聚类、有重叠聚类和模糊聚类等。采用 k-均值、k-中心点等算法的聚类分析工具已被加入到许多著名的统计分析软件包中，如SPSS、SAS等。

聚类是搜索簇的无监督学习过程。与分类不同，无监督学习不依赖预先定义的类或带类标记的训练实例，需要由聚类学习算法自动确定标记，而分类学习的实例或数据对象有类别标记。

从实际应用的角度来看，聚类分析是数据挖掘的主要任务之一。聚类能够作为一个独立的工具获得数据的分布状况，观察每一簇数据的特征，集中对特定的聚簇集合作进一步的分析。聚类分析还可以作为其他算法（如分类和定性归纳算法）的预处理步骤。

1．聚类分析的定义

聚类分析是依据研究对象（样品或指标）的特征，对其进行分类的方法，减少研究对象的数目。各类事物缺乏可靠的历史资料，无法确定共有多少类别，目的是将性质相近的事物归入一类。各指标之间具有一定的相关关系。聚类分析是一组将研究对象分为相对同质的群组的统计分析技术。

2．聚类方法

（1）层次聚类（Hierarchical Clustering）

层次聚类方法分为合并法、分解法和树状图。

（2）非层次聚类

非层次聚类分为划分聚类和谱聚类。

SAS 系统常用的聚类过程有 CLUSTER 过程、FASTCLUS 过程、TREE 过程和 VARCLUS 过程。

（1）CLUSTER 过程

CLUSTER 过程属于对指标进行系统聚类，对应原始数据以欧氏距离为默认的距离计算方法。对于观测值之间的距离先用 distance 过程将数据转换成相应的距离数据在聚类。CLUSTER 过程的结果输出到数据集后可以供 TREE 过程来显示树图，以便更形象地显示聚类过程。

```
语法格式：PROC    cluster method=聚类方法    <选项列表>;
                  var    聚类用变量;
                  copy  复制变量;
                  id  变量名;
                  freq  变量;
          RUN;
```

【语法解读】

1）method=聚类方法：指定聚类分析的方法，属于必须项。常用聚类方法如下。

average|avg：平均连接法，若不指定 nosquare，距离数据被平方。

Ward|war：最小方差法，若不指定 nosquare，距离数据被平方。

eml：最大似然法，对球形多元正态分布混合进行最大似然估计。

Centroid|cen：重心法，若不指定 nosquare，距离数据被平方。

Mcquity|mcq：加权平均连接法。

2）<选项列表>：常用选项如下。

Data=：指定要聚类的数据集。

Outtree=：创建一个树状数据集。TREE 过程可以把 CLUSTER 过程产生的 OUTTREE = 数据集作为输入，画出聚类谱系图，并按照用户指定的聚类水平（类数）产生分类结果数据集。

P=：指定系统聚类过程的代数，默认为显示所有代数，如果指定 p=0 会抑制聚类过程的显示。

Nosquare：阻止输入数据被平方。

3）var：指定用来进行聚类分析的变量，省略此语句时则默认为没有出现在其他语句中的所有数值型变量。一般 cluster 过程和 var 语句是必需的，其余都为可选。

4）copy <复制变量>：此语句指定的变量将被复制到 outtree=语句指定的数据集中。

5）id <变量名>：对应 outtree=语句的数据集，用 ID 变量值作为标示观测。

6）freq 变量：指定频数变量。

（2）FASTCLUS 过程

FASTCLUS 过程称为动态聚类过程，又称为快速聚类，主要用于大样本数据的聚类。变量之间以欧氏距离为基础对数据进行分析，可以指定分类的最大数目。此聚类过程以迭代思想为理论基础，先对样本观测粗略分类，然后按某种最优准则逐步修改分类至最优为止，其适用于观测量大的数据。在 FASTCLUS 过程中必须指定 maxclusters=（最大凝聚点）或 randius=

（指定凝聚点间最短距离）选项。

语法格式：PROC fastclus maxclusters=n|randius=t <选项列表>;

```
var  聚类用变量;
by  变量表;
id  变量名;
freq  变量;
weight  变量;
RUN;
```

【语法解读】

1）maxclusters=n|：指定最大“凝聚点”数。

2）randius=t：指定凝聚点间的最短距离。

3）weight 变量：指定权重分析变量。

（3）VARCLUS 过程

VARCLUS 过程对一组数值变量进行系统聚类，聚类的选择是使每一类的第一主成分或者重心分量所解释的变异为最大，属于一种减少变量的方法聚类。

语法格式：PROC varclus <选项列表>;

```
var  聚类用变量;
seed  变量表;
partial  变量表;
by  变量表;
id  变量名;
freq  变量;
weight  变量;
RUN;
```

【语法解读】

1）<选项列表>：常用选项如下。

Data=：指定要聚类的数据集。

Outtree=：创建一个树状数据集。TREE 过程可以把 CLUSTER 过程产生的 OUTTREE = 数据集作为输入，画出聚类谱系图，并按照用户指定的聚类水平（类数）产生分类结果数据集。

控制聚类常用方法选项如下。

Cov：分析协方差矩阵。

Seed：将 seed 语句中列出的变量作为一个类的基准分类。

Hi：不同层次上聚类构成一个分层体系结构聚类。

控制聚类数常用选项如下。

Minc=n：指定最小分类数，默认为 2。

Max=n：指定最大分类数，默认值为变量数。

控制显示输出的项如下。

Noprint：不打印输出结果到输出窗口。

Trace：跟踪每个变量被分类的过程。

Summary：给出最后的总结表，其他输出被抑制。

2）seed 变量表：指定用来作为初始分类凝聚点的变量。若选项了 seed 语句，不必再有 initial=seed 选项；如果指定了 initial=s 选项，则 seed 语句无效。

3）partial 变量表：指定偏相关聚类分析变量。

（4）TREE 过程

TREE 过程是对 CLUSTER 聚类过程或 VARCLUS 聚类过程创建的数据集显示树图，让聚类的结果更加形象。

语法格式：

```
PROC tree <选项列表>;
        Name 变量表;
     height   变量;
     id 变量;
     freq 变量;
      by 变量表;
RUN;
```

【语法解读】

1）<选项列表>：常用可取选项如下。

Data=：指定树的输入数据集。

Out=：指定输出数据集。

H=变量：指定某个变量作为树高度。

Level=n：指定树的水平。

2）Name 变量表：指定一个字符或者数值变量，标示每个观测节点。

3）height 变量：指定一个数值变量，定义树中每个节点的高度。

7.6.2 聚类分析应用

聚类分析的应用领域比较广泛。它是研究分类问题的一种多元统计分析方法。

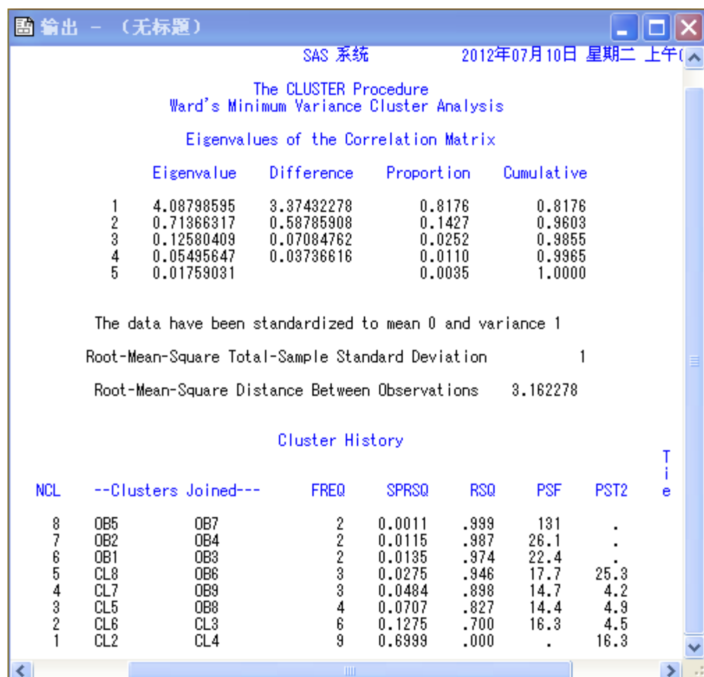
【例 7.15】 表 7-14 所示是全国 9 省市居民 2011 年支出情况数据汇总资料，主要涉及生活消费支出情况的 8 个指标。

表 7-14 全国 9 省市居民 2011 年支出情况数据

地　区	食品消费	居　住	医疗花费	交通和通信	教　育
天津	1 117.72	1 200.16	6 000.10	800.32	6 800.87
北京	2 300.12	1 600.88	7 898.92	1 300.89	12 000.56
吉林	1 020.00	780.08	5 456.21	678.21	4 000.32
上海	2 287.15	1 889.23	8 356.21	1 500.23	15 000.21
江苏	1 317.88	467.62	163.16	293.07	6 700.21
浙江	1 838.57	798.88	326.12	496.86	8 900.96
福建	1 408.54	430.14	136.40	306.06	7 680.09
山东	1 100.13	560.97	1 678.85	221.93	298.23
广东	1 681.68	1 700.21	8 700.19	900.23	12 000.26

```
data   diaocha;
  input   diqu   $   shipin   house   yiliao   jiaotong   education;
cards;
天津  1117.72 1200.16 6000.10 800.32 6800.87
北京  2300.12 1600.88 7898.92 1300.89 12000.56
吉林  1020.00 780.08 5456.21 678.21 4000.32
上海  2287.15 1889.23 8356.21 1500.23 15000.21
江苏  1317.88 467.62 163.16 293.07 6700.21
浙江  1838.57 798.88 326.12 496.86 8900.96
福建  1408.54 430.14 136.40 306.06 7680.09
山东  1100.13 560.97 1678.85 221.93 298.23
广东  1681.68 1700.21 8700.19 900.23 12000.26
;
run;
proc   cluster   data =diaocha   standard   method =ward
       outtree =jltree   pseudo;
       copy   diqu;
run;
proc   tree   data =jltree   horizontal;
     id   diqu;
run;
```

程序执行后输出窗口显示如图 7-24 和图 7-25 所示。



输出 - （无标题）

SAS 系统　　2012年07月10日 星期二 上午(

The CLUSTER Procedure
Ward's Minimum Variance Cluster Analysis

Eigenvalues of the Correlation Matrix

	Eigenvalue	Difference	Proportion	Cumulative
1	4.08798595	3.37432278	0.8176	0.8176
2	0.71366317	0.58785908	0.1427	0.9603
3	0.12580409	0.07084762	0.0252	0.9855
4	0.05495647	0.03736616	0.0110	0.9965
5	0.01759031		0.0035	1.0000

The data have been standardized to mean 0 and variance 1

Root-Mean-Square Total-Sample Standard Deviation　　1

Root-Mean-Square Distance Between Observations　　3.162278

Cluster History

NCL	--Clusters Joined---		FREQ	SPRSQ	RSQ	PSF	PST2	Tie
8	OB5	OB7	2	0.0011	.999	131	.	
7	OB2	OB4	2	0.0115	.987	26.1	.	
6	OB1	OB3	2	0.0135	.974	22.4	.	
5	CL8	OB6	3	0.0275	.946	17.7	25.3	
4	CL7	OB9	3	0.0484	.898	14.7	4.2	
3	CL5	OB8	4	0.0707	.827	14.4	4.9	
2	CL6	CL3	6	0.1275	.700	16.3	4.5	
1	CL2	CL4	9	0.6999	.000	.	16.3	

图 7-24　CLUSTER 聚类分析

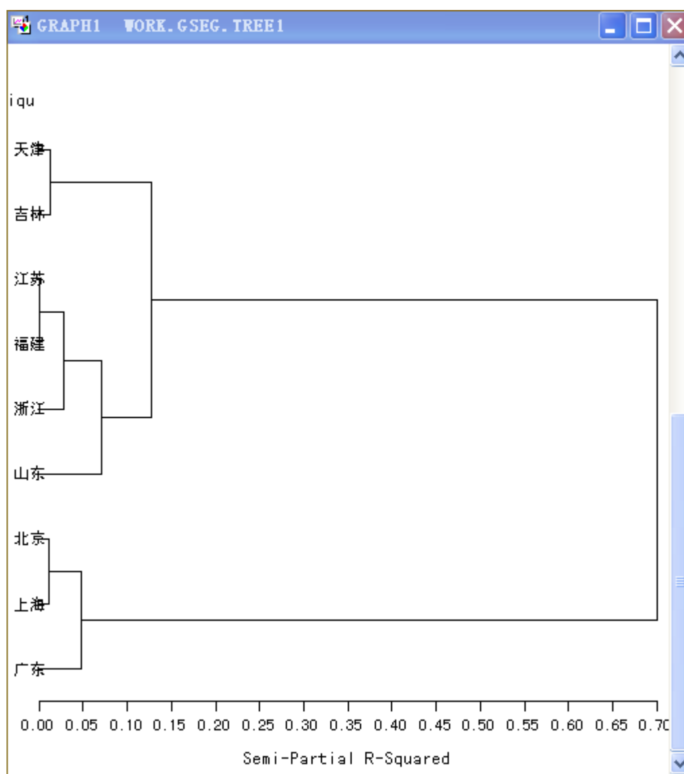


图 7-25　聚类分析树

【程序解读】

1）PROC cluster data =diaocha standard method =ward outtree =jltree　pseudo：method =ward 指定选择 ward 方法聚类，pseudo 项显示为 F 及 t 的平方统计量。该选项只有当数据是坐标或 method=ward、average、centroid 才有效。

2）copy diqu：将变量 diqu 复制到 outtree=指定的数据集中。

3）PROC tree data =jltree　horizontal：horizontal 项指定树的高度轴为水平方向，默认为垂直方向。

4）ID diqu：指定树图中识别对象。

【分析结果解读】

由图 7-24 中的 PSF 列可以看出 G=2 和 G=3 处有峰值，因此分为 2 类或 3 类最好。由图 7-25 可以看出 RSQ 变化最大，从 0.946 变化到 0.7，可以看出分为 2 类最好。

7.7　判别分析基础

判别分析（Discriminant Analysis）是一种进行统计鉴别和分组的技术手段，是根据观测到的某些指标对研究对象进行分类的一种统计方法。

7.7.1　判别分析概述

判别分析（Discriminant Analysis）是一种进行统计鉴别和分组的技术手段。判别分析是

根据已知样本的分类以及所测的指标，筛选出能够提供较多信息的指标，从而建立判别方程，使其错判率最小的一种方法。根据判别方程，将未知分类的样本指标带入判别方程，从而判断它属于哪个总体。

基本思想：根据已知类别的样本所提供的信息，总结出分类的规律性，建立判别公式和判别准则，根据判别函数判别新的样本所属类型。

常用判别方法如下：

（1）距离判别法

距离判别最直观的方法是根据各样本与母体之间的距离远近做出判别属于哪一类，哪个距离最小就将它判归哪个总体。

（2）Fisher 判别法

Fisher 判别法的基本思想是投影，即将原来在 R 维空间的自变量组合投影到维度较低的 D 维空间去，然后在 D 维空间再进行分类。投影的原则是使得每一类内的离差尽可能小，而不同类内的离差尽可能大。

（3）贝叶斯判别法（Bayes）

贝叶斯判别法是根据先验概率求出后验概率，并根据后验概率分布给出统计推断，所有 g 个类别都是空间中互斥的子域，每个观测值都是空间中的一个点，在考虑先验概率的情况下利用 Baye 公式按照一定准则构造一个判别函数，分别计算该样本落入各个子域的概率，所有概率中最大的一类就被认为是该样本所属的类别。

（4）逐步判别

对于一些变量使用判别并没有什么作用，为了得到对判别最合适的变量，可以使用逐步判别。也就是，一边判别，一边引进判别能力最强的变量（通过假设检验的方法）。

SAS 系统常用的判别分析过程有 DISCRIM 过程和 STEPDISC 过程。

1．DISCRIM过程

DISCRIM 过程可以对一个或多个数值变量计算线性或二次判别函数，对观测进行分类。此过程涉及两种判别方法，参数判别法和非参数判别法。

语法格式：
```
PROC discrim <选项>;
        class 变量;
        by 变量表;
        freq 变量;
        id 变量;
        priors 概率表;
        testclass 变量;
        testfreq 变量;
        testid 变量;
        var 变量表;
        weight 变量;
RUN;
```

【语法解读】

1）DISCRIM 过程语句中常用<选项>如下。

① 数据集选项如下。

Data=数据集名：指定判别分析的数据集名。

TESTDATA=数据集名：指定分类的原始数据集名。

outstat=输出数据集名：指定分析输出数据集选项，数据集中包括均值、标准差、相关系数等多种统计量。

Out=数据集名：指定输出数据集，包括 data=数据集中的所有观测，以及每个观测的后验概率和由回代法判别后的类别。

outcross=数据集名：指定输出数据集，包括 data=数据集中的所有观测，以及每个观测的后验概率和由交叉验证法判别后的类别。

testout=数据集名：生成一个输出数据集，包含来自 TESTDATA=数据集的所有数据，以及后验概率和每个观测通过交叉确认被分入的类。

Outd=数据集名：指定输出数据集包括 data=数据集中的所有观测，以及每个观测指定组的密度估计。

② 选择判别分析类型的选项如下。

METHOD=NORMAL：指定 METHOD=NORMAL 时，基于类内服从多元正态分布，并导出线性或二次判别函数。默认值为 NORMAL。

METHOD=NPAR：指定 METHOD=NPAR 时，属于非参数法，对观测进行分类，估计指定组的密度。

POOL=NO|TEST|YES：确定平方距离的度量是以合并协方差阵还是组内协方差阵为基础。默认值为 POOL=YES。当 POOL=YES 时，采用合并协方差阵得出线性判别函数；当 POOL=NO 时，采用单个组内协方差阵得出二次判别函数，当 METHOD=NORMAL 时，POOL=TEST 要求对组内协方差阵的齐性的似然比检验进行 Bartlet 修正。

SLPOOL=P：指定齐次性检验的统计意义水平。默认值为 0.10。

③ 指定判别标准选项如下。

THRESHOLD=P：分类时可以接受的最小后验概率（$0<p<1$），通过此选项控制成员观测分配，对于组成员的最大后验概率小于 THRESHOLD 指定值的观测被分配到 other 组中。默认值为 0。

④ 控制显示统计信息选项如下。

ALL：所有控制显示输出选项。

SHORT：取消显示某些默认的输出选项。

NOPRINT：不打印显示结果到输出窗口。

⑤ 相似相关矩阵选项如下。

PCORR：以合并的类内相关矩阵显示。

BCORR：以类间相关矩阵显示。

TCORR：以全体样本相关矩阵显示。

WCORR：对每个分类水平以类内相关矩阵显示。

⑥ 交叉确认分类选项如下。

CROSSLIST：对每个观测打印交叉确认分类结果。

CROSSLISTERROR：只对错误分类的观测打印交叉确认分类结果。

⑦ 检验信息和统计量选项如下。

MANOVA：输出显示检验总体中假设各类均值相等的多元方差分析。

ANOVA：输出显示检验各类假设每个变量总体均值相等的单变量方差分析。

SIMPLE：显示简单统计描述信息。

STDMEAN：输出显示整体样本和合并的各类内标准化分类均值。

2）priors 概率表：语句指定各组中成员出现的先验概率。3 种语句如下。

PRIORS EQUAL：规定先验概率相等。

PRIORS PROPORTIONAL|PROP：先验概率正比于样本容量。

PRIORS probabilities：对分类变量的每个水平定义一个先验概率。

3）testclass 变量：命令 testdata=数据集中的一个变量，以确定该数据集中观测是否被错分。

4）testfreq 变量：对 testdata=选项指定的数据集变量求出观测的频度。

5）testid 变量：testid 语句声明仅在语句中出现 testlist 或 testlisterr 选项时才有效。显示 testdata 数据集的分类结果时，testid 中变量值将取代每个观测序号。

2．STEPDISC过程

STEPDISC 过程称为逐步判别过程分析，此过程通过向前选入、向后剔除或逐步选择对判别有用的定量变量来完成逐步判别分析。变量根据以下 3 条准则之一选入或剔除模型：

（1）向前选入法

开始时模型中没有变量，每一步贡献度的最大进入模型。当不再有未被选入的变量小于选入的临界值时，向前选入过程停止。

（2）向后剔除

开始时，所有变量依赖于 VAR 语句中的变量都在模型中。每一步判断贡献度最小的准则下对模型中判别能力贡献最小的变量剔除。当所有余下的变量都达到留在模型中的标准时，向后剔除过程停止。

（3）逐步选择

开始时如同向前选择一样，模型中没有变量，每一步都被检查。在贡献度准则下统计量对模型的判别能力贡献最小的变量达不到留在模型中的标准，它就被剔除。否则，不在模型中对模型的判别能力贡献最大的变量被选入模型。当模型中的所有变量都达到留在模型中的标准而没有其他变量能达到进入模型的标准时，逐步选择过程停止。

语法格式：

STEPDISC 过程的语句：

```
PROC stepdisc <选项>;
    class 变量;
    by 变量表;
    freq 变量;
    var 变量表;
    weight 变量;
RUN;
```

【语法解读】

1）stepdisc <选项>：该语句中具有的常用选项如下。

data=数据集名：指定进行判别分析的数据集。

method=FW/BW/SW：指定选择模型中变量的方法：FORWARD（FW），BACKWARD（BW）

和 STEPWISE（SW）。

SLENTRY= |SLE= ：在向前选择方法中，指定选入变量的显著性水平。

SLSTAY= |SLS= ：在向后剔除方法中，指定保留变量的显著性水平。默认值为 0.15。

PR2ENTRY＝p|PR2E＝p：在向前选择方式中，指定选入变量的偏。

PR2STAV＝p|PR2S＝p：在向后剔除方式中，指定保留变量的偏。

Include=n：对 var 语句中指定的变量，前 n 个变量总包含在模型中。默认值为 0。

Maxstep=：指定选择变量最大的步数。默认值为 var 语句中指定的变量个数的两倍。

Stop=：指定最后模型中保留的变量个数。当模型变量中的个数为 stop=参数指定的变量个数时，停止变量选择。

Simple：简单统计量。

tcorr：全样本相关。

wcorr：类内水平相关。

Stdmean：标准化类均值。

2）class 变量：定义分析组变量。

7.7.2 判别分析应用

在医学研究和疾病防治工作中，经常会遇到需要根据观测到的资料对所研究的对象进行分类的问题。

【例 7.16】 农业样本抽样检验，对病菌蔬菜用 1 标识，非病菌蔬菜用 2 标识，两类样本各抽样 5 例化验 4 项指标，用逐步判别法对 10 个样品进行判别归类。

```
data    vegetable;
 input    group    v1-v4;
cards;
1 228 134 20 11
1 245 134 10 40
1 200 167 12 27
1 170 150 7 8
1 100 167 20 14
2 185 115 5 19
2 170 125 6 4
2 165 142 5 3
2 135 108 2 12
2 100 117 7 2
;
run;
proc    stepdisc    data=vegetable;
/* proc stepdisc 逐步判别进行变量删选*/
   class    group; /*分组变量*/
     var    v1-v4;/*分析变量*/
run;
/*输出结果保留变量集为{v2,v3,v4}*/
proc    discrim    data=vegetable list;
```

```
    class   group;/*分组变量*/
    var   v2   v3   v4;/*根据 STEPDISC 过程选择出的变量进行变量分析*/
run;
```

程序执行后输出窗口显示如图 7-26～图 7-31 所示。

输出 - （无标题）

SAS 系统　　2012年07月13日 星期五 上午

The STEPDISC Procedure

The Method for Selecting Variables is STEPWISE

Total Sample Size	10	Variable(s) in the Analysis	4
Class Levels	2	Variable(s) Will Be Included	0
		Significance Level to Enter	0.15
		Significance Level to Stay	0.15

Number of Observations Read	10
Number of Observations Used	10

Class Level Information

group	Variable Name	Frequency	Weight	Proportion
1	_1	5	5.0000	0.500000
2	_2	5	5.0000	0.500000

图 7-26　逐步判别法选择变量分析

输出 - （无标题）

SAS 系统　　2012年07月13日 星期五 上午08时52分57秒　3

The STEPDISC Procedure

Stepwise Selection Summary

Step	Number In	Entered	Removed	Partial R-Square	F Value	Pr > F	Wilks' Lambda	Pr < Lambda	Average Squared Canonical Correlation	Pr > ASCC
1	1	v3		0.5557	10.01	0.0133	0.44431688	0.0133	0.55568312	0.0133
2	2	v4		0.3353	3.53	0.1023	0.29532778	0.0140	0.70467222	0.0140
3	3	v2		0.3241	2.88	0.1407	0.19959873	0.0160	0.80040127	0.0160

图 7-27　逐步判别法选择变量信息

输出 - （无标题）

SAS 系统　　2012年07月13日 星期五 上

The DISCRIM Procedure

Total Sample Size	10	DF Total	9
Variables	3	DF Within Classes	8
Classes	2	DF Between Classes	1

Number of Observations Read	10
Number of Observations Used	10

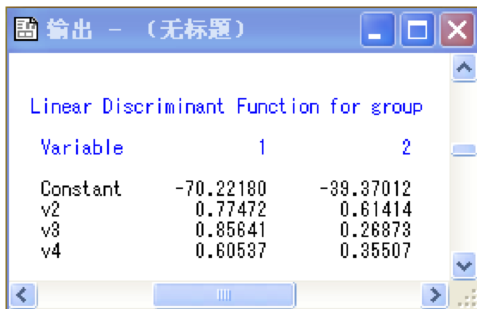
Class Level Information

group	Variable Name	Frequency	Weight	Proportion	Prior Probability
1	_1	5	5.0000	0.500000	0.500000
2	_2	5	5.0000	0.500000	0.500000

Pooled Covariance Matrix Information

Covariance Matrix Rank	Natural Log of the Determinant of the Covariance Matrix
3	12.89623

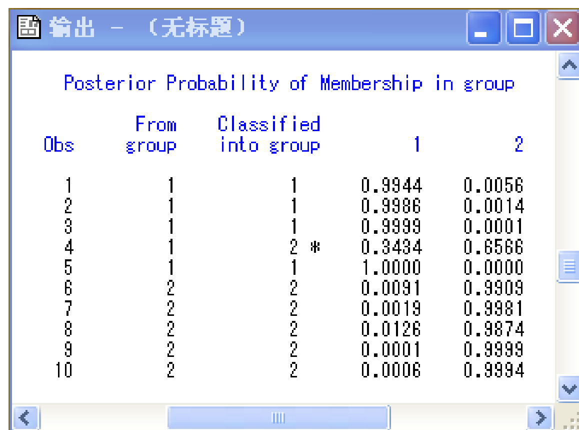
图 7-28　判别分析过程 DISCRIM 分析（一）



Linear Discriminant Function for group

Variable	1	2
Constant	-70.22180	-39.37012
v2	0.77472	0.61414
v3	0.85641	0.26873
v4	0.60537	0.35507

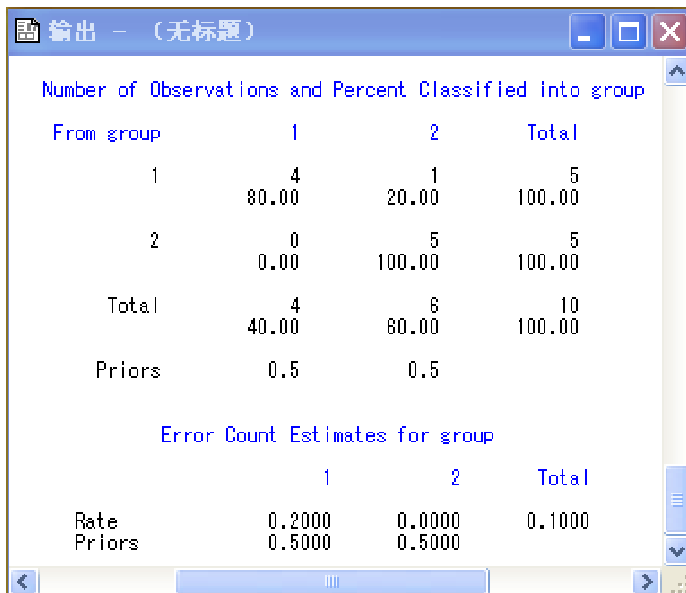
图 7-29　判别分析过程 DISCRIM 分析（二）



Posterior Probability of Membership in group

Obs	From group	Classified into group	1	2
1	1	1	0.9944	0.0056
2	1	1	0.9986	0.0014
3	1	1	0.9999	0.0001
4	1	2 *	0.3434	0.6566
5	1	1	1.0000	0.0000
6	2	2	0.0091	0.9909
7	2	2	0.0019	0.9981
8	2	2	0.0126	0.9874
9	2	2	0.0001	0.9999
10	2	2	0.0006	0.9994

图 7-30　判别分析过程 DISCRIM 分析（三）



Number of Observations and Percent Classified into group

From group	1	2	Total
1	4 80.00	1 20.00	5 100.00
2	0 0.00	5 100.00	5 100.00
Total	4 40.00	6 60.00	10 100.00
Priors	0.5	0.5	

Error Count Estimates for group

	1	2	Total
Rate	0.2000	0.0000	0.1000
Priors	0.5000	0.5000	

图 7-31　分析过程 DISCRIM 分析（四）

【程序解读】

1）PROC stepdisc data=vegetable：proc stepdisc 逐步判别进行变量删选，data=指定要进行逐步判别分析的数据集为 vegetable。

2）class group：指定分组变量。

3）var v1-v4：var 语句指定分析变量为 v1、v2、v3 和 v4。

4）PROC discrim data=vegetable list：调用判别分析过程 DISCRIM，list 选项列出详细信息。

5）var v2 v3 v4：根据 stepdisc 过程选择出的变量进行变量分析。

【分析结果解读】

根据 STEPDISC 逐步判别进行变量删选，由图 7-26 和图 7-27 可以看出变量 v2、v3 和

v4 被选择进入分析队列，然后运用判别分析过程 DISCRIM 对 3 个指标变量 v2、v3 和 v4 进行分析。可以看出这 3 个变量对蔬菜病菌影响起到了主要的作用，应严格对这 3 个变量指标进行跟踪。

7.8 客户流失分析案例

随着外资银行和国内信用卡行业的发展，国外和国内行业内的竞争压力与日俱增，信用风险的管理和客户特征的研究也就显得越来越重要，而较高的客户流失率也逐渐成为管理者必须面对和认真对待的一个问题。

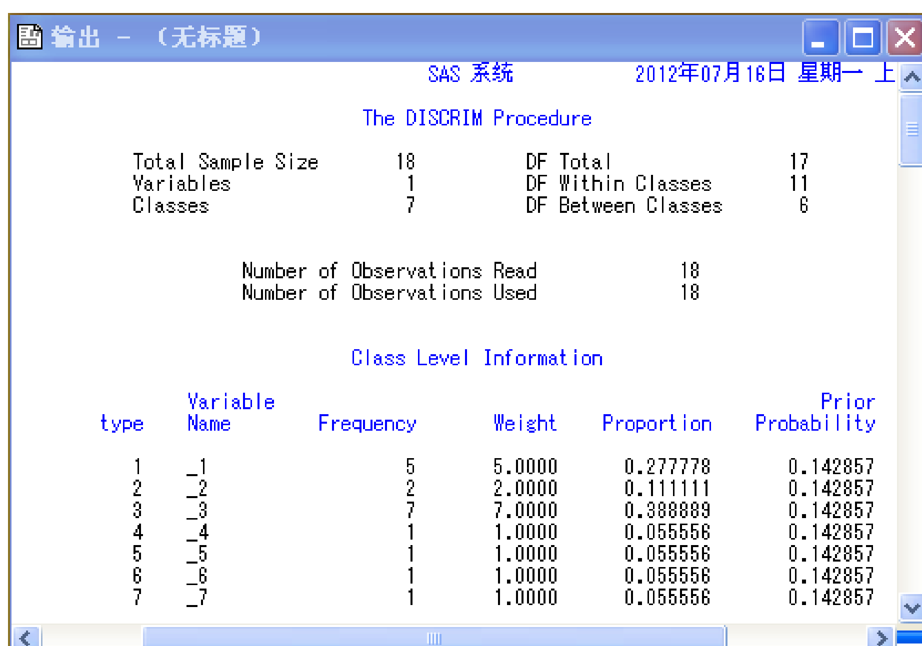
【例 7.17】 根据信用卡客户数据分析客户流失原因。

表 7-15 客户流失原因表

原 因	标 识
客服服务态度差	1
额度低	2
还款不方便	3
人工服务不到位，接入需要很长时间	4
年费高	5
积分面值低	6
调额度困难	7

```
*对外部数据处理;
%let   path= D:\jx\lius;   /*定义外部文件路径*/
%let   type=.txt;
%let   fil= "&path&type";
libname   jx 'd:\jx';   /*定义逻辑库*/
data   jx.lius;   /*数据集存储到指定逻辑库*/
       Infile   &fil   dlm='|'   dsd   missover;
              input                  bh      :$4.
                                      type    :1.
                                      times   :3.
                     ;
run;
proc   discrim   data=jx.lius list;
     class   type;/*分组变量为投诉类型*/
     var     times;/*对 times 投诉次数分析*/
run;
```

程序执行后输出窗口显示如图 7-32～图 7-34 所示。



输出 － （无标题）

SAS 系统　　2012年07月16日 星期一 上

The DISCRIM Procedure

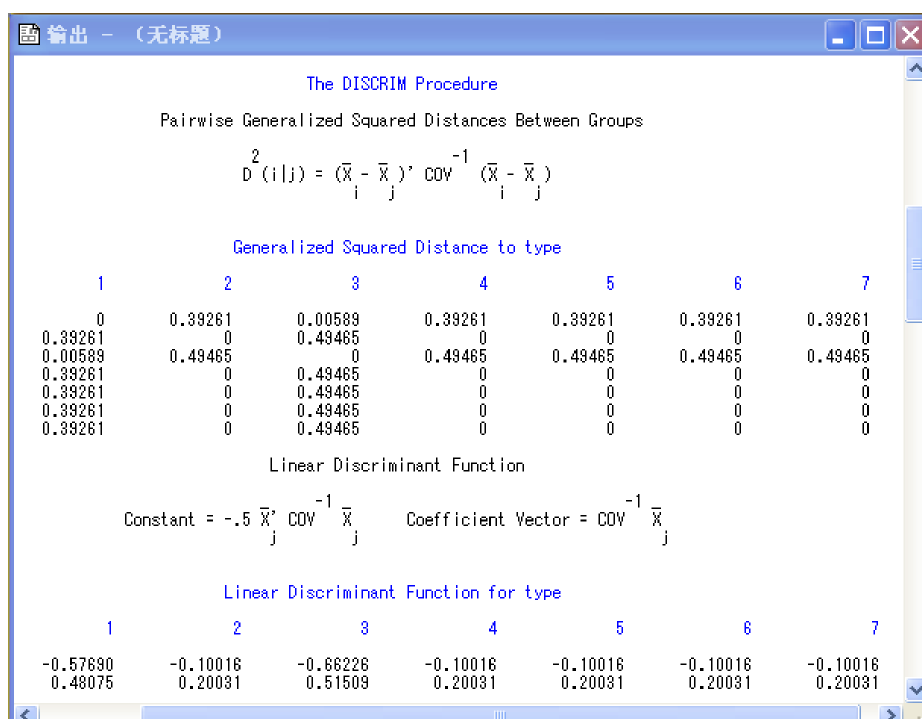
Total Sample Size	18	DF Total	17
Variables	1	DF Within Classes	11
Classes	7	DF Between Classes	6

Number of Observations Read	18
Number of Observations Used	18

Class Level Information

type	Variable Name	Frequency	Weight	Proportion	Prior Probability
1	_1	5	5.0000	0.277778	0.142857
2	_2	2	2.0000	0.111111	0.142857
3	_3	7	7.0000	0.388889	0.142857
4	_4	1	1.0000	0.055556	0.142857
5	_5	1	1.0000	0.055556	0.142857
6	_6	1	1.0000	0.055556	0.142857
7	_7	1	1.0000	0.055556	0.142857

图 7-32　判别分析类信息



输出 － （无标题）

The DISCRIM Procedure

Pairwise Generalized Squared Distances Between Groups

$$D^2(i|j) = (\bar{X}_i - \bar{X}_j)' \, COV^{-1} \, (\bar{X}_i - \bar{X}_j)$$

Generalized Squared Distance to type

1	2	3	4	5	6	7
0	0.39261	0.00589	0.39261	0.39261	0.39261	0.39261
0.39261	0	0.49465	0	0	0	0
0.00589	0.49465	0	0.49465	0.49465	0.49465	0.49465
0.39261	0	0.49465	0	0	0	0
0.39261	0	0.49465	0	0	0	0
0.39261	0	0.49465	0	0	0	0
0.39261	0	0.49465	0	0	0	0

Linear Discriminant Function

$$\text{Constant} = -.5 \, \bar{X}_j' \, COV^{-1} \, \bar{X}_j \qquad \text{Coefficient Vector} = COV^{-1} \, \bar{X}_j$$

Linear Discriminant Function for type

1	2	3	4	5	6	7
-0.57690	-0.10016	-0.66226	-0.10016	-0.10016	-0.10016	-0.10016
0.48075	0.20031	0.51509	0.20031	0.20031	0.20031	0.20031

图 7-33　判别分析函数分析应用

输出 - （无标题）

Number of Observations and Percent Classified into type

From type	1	2	3	4	5	6	7	Other	Total
1	0 0.00	0 0.00	2 40.00	0 0.00	0 0.00	0 0.00	0 0.00	3 60.00	5 100.00
2	0 0.00	0 0.00	0 0.00	0 0.00	0 0.00	0 0.00	0 0.00	2 100.00	2 100.00
3	1 14.29	0 0.00	2 28.57	0 0.00	0 0.00	0 0.00	0 0.00	4 57.14	7 100.00
4	0 0.00	0 0.00	0 0.00	0 0.00	0 0.00	0 0.00	0 0.00	1 100.00	1 100.00
5	0 0.00	0 0.00	0 0.00	0 0.00	0 0.00	0 0.00	0 0.00	1 100.00	1 100.00
6	0 0.00	0 0.00	0 0.00	0 0.00	0 0.00	0 0.00	0 0.00	1 100.00	1 100.00
7	0 0.00	0 0.00	0 0.00	0 0.00	0 0.00	0 0.00	0 0.00	1 100.00	1 100.00
Total	1 5.56	0 0.00	4 22.22	0 0.00	0 0.00	0 0.00	0 0.00	13 72.22	18 100.00
Priors	0.14286	0.14286	0.14286	0.14286	0.14286	0.14286	0.14286		

Error Count Estimates for type

	1	2	3	4	5	6	7	Total
Rate	1.0000	1.0000	0.7143	1.0000	1.0000	1.0000	1.0000	0.9592
Priors	0.1429	0.1429	0.1429	0.1429	0.1429	0.1429	0.1429	

图 7-34　判别分析各类型观测值与占比

【程序解读】

1）class type：指定分组变量为投诉类型。

2）var times：对 times 投诉次数分析。

【分析结果解读】

根据收集的信息卡数据进行分析，判别分析类信息类型为 1 和 3 的频率比较高，如图 7-32 所示。判别分析函数分析应用可以得到判别函数式，如图 7-33 所示。通过图 7-34 可以看出类型 1 和 3 为客户流失的主要原因，因此只有对客服服务态度差和还款不方便两个因素采取措施，才能挽留住客户。

第 8 章　SAS与关系数据库Oracle交互应用

8.1　SAS与Oracle交互基础

SAS 与关系数据库的连接体现了 SAS 与其他数据库的交互应用能力，通过 SAS/Access 模块建立与关系型数据库的连接，提取关系数据库数据，生成 SAS 能够识别的数据集，为 SAS 其他模块的应用提供数据支撑，建立了连接关系数据库的通道。业界应用中大量的数据存储在数据库库中，Oracle 数据库是大型数据库之一。数据仓库提供数据支持，SAS 对数据仓库中的数据进行分析、数据挖掘、报表开发等。

8.1.1　SAS与Oracle数据库连接概述

SAS 通过 SAS/Access 模块建立与关系数据库的连接，通过该模块可以读取关系数据库里的数据，生成 SAS 语言能够识别的数据集，以便其他模块运用关系数据库数据进行分析、数据挖掘等。SAS 语言是通过逻辑库方式连接 SAS 与 Oracle 的数据库，此处的逻辑库就相当于在河上面建的一座桥，河的一边是 SAS 系统，另一边是关系数据库，通过逻辑库这座桥建立 SAS 和关系数据库互相通信的通道。

为帮助读者理解 SAS 与关系数据库如何建立连接通道，本章通过关系数据库 Oracle 来进行讲解。Oracle 是应用比较广泛的关系型数据库，在大数据量存储中作为数据仓库的基础数据库，其功能强大。表可以通过 SAS 逻辑库来指向 Oracle 数据库中的表，这样就建立了一个和数据库连接的通道。

Oracle 数据库中的数据存储形式是一张关系二维表，如图 8-1 所示。

	ID	NAME	ENGLISH	CHINESE	MATH
1	10002	马小明	68	75	36
2	10001	刘小红	78	89	90

图 8-1　Oracle 数据库表存储数据形式

【例 8.1】 读取 Oracle 数据库中的一张 custinf 表，建立数据集 custin，登录数据库的用户名 user=chiran，密码 password=chiran，数据库实例 path=orcl。

程序如下：

```
libname   jx   oracle   user=chiran   password=chiran   path=orcl;
data   custinf;
       set   jx.custinf;   /*set 语句读取数据库中的表，逻辑库名.数据库中的表名*/
run;
```

【程序解读】

1）libname jx oracle user=chiran password=chiran path=orcl;：数据库引擎为 Oracle，user=用户

名，password=密码，path=数据库实例。

2）set jx.custinf;：set 语句读取数据，jx 是连接 Oracle 数据库的逻辑库名，custinf 为 Oracle 关系数据库中的表名。

8.1.2 SAS获取Oracle数据

SAS 获取其他关系数据库数据首先要建立通信通道，SAS 语言建立与 Oracle 数据库的连接方式的通信通道可以通过创建逻辑库或定义宏变量两种方式。

（1）逻辑库连接数据库语法

LIBNAME 逻辑库名 ORACLE USER=登录数据库用户名 PASSWORD=登录数据库密码 PATH=数据库实例;

【例 8.2】建立逻辑库名为 test，连接 Oracle 数据库，Oracle 数据库登录的用户名为 chiran，密码为 chiran，数据库实例为 orcl。

```
libname  test  oracle user=chiran  password=chiran  path=orcl;
```

【程序解读】

test 为逻辑库名，Oracle 为数据库引擎，user=chiran 为指定连接关系数据库的用户名，password=chiran 为指定连接关系数据库的密码，path=orcl 为指定连接关系数据库的实例名。

（2）定义宏变量连接数据库语法

%let 宏变量名=user=登录数据库用户名 password=登录数据库密码 path=数据库实例;

【例 8.3】对例 8.2 通过定义宏变量的方式建立关系数据库连接通道，宏变量名为 test_cnt。

```
%let   test_cnt=user=chiran  password=chiran  path=orcl;
/*定义宏变量，建立与数据库连接的变量*/
```

【程序解读】

%let：定义宏变量语句关键字，test_cnt：定义的宏变量名。

【例 8.4】 客户销售数据存储在 Oracle 数据库表 sale 中，请取出北京市的数据，生成数据集 sale_beijing。

该实验的具体步骤如下：

1）首先在 Oracle 数据库中创建表 sale，创建表语句如下：

```
-- Create table
create  table  sale
(
  empno      NUMBER(4) not null,
  diqu       VARCHAR2(10),
  sal        NUMBER(7,2),
  deptno     NUMBER(2)
)
Tablespace  users;
```

2）向表 sale 插入数据，脚本如下：

```
Insert   into   sale   values(1001,'北京市',12876.23,10);
Insert   into   sale   values(1002,'上海市',26872.87,12);
Insert   into   sale   values(1003,'天津市',39871.86,13);
Insert   into   sale   values(1004,'山东省',55873.62,15);
Insert   into   sale values(1005,'河北省',65879.29,16);
commit;
```

3）SAS 程序如下：

```
*SAS 程序提取 Oracle 数据库表 Sale 中销售地区为北京的数据，生成 SAS 数据集 sale_beijing;
libname    jx   oracle   user=chiran   password=chiran   path=orcl;
      /*建立与 Oracle 数据库连接的逻辑库*/
libname   jxsj   'd:\jx';/*数据集存储逻辑库*/
data   jxsj.sale_beijing;
set   jx.sale   (where=(diqu LIKE '北京%'));
/*读数据库中的表 sale，把北京地区的读取出来*/
run;
```

【程序解读】

set jx.sale (where=(diqu LIKE '北京%'));：读取 Oracle 数据库表 sale，where 语句为过滤条件，取 diqu 字段，通过模式匹配 like 语句取出“北京”地区的信息。

【例 8.5】 通过 SQL 过程，向 Oracle 数据库表 sale 插入一条记录。

```
 *向 ORACLE 数据库表插入数据;
libname   jx   ORACLE   user=chiran   password=chiran   path=orcl;
 /*建立逻辑库，连接 Oracle 数据库*/
*调用 SQL 过程;
proc   sql   noprint ;   /*调用过程 SQL 进行处理*/
insert   into   jx.sale   (empno,diqu,sal,deptno)   values(1006,'广东省',51233.67,19);
/*通过 insert   into 语句向 Oracle 数据库插入记录*/
quit; /*结束程序*/
```

【程序解读】

通过调用 SQL 过程，SAS 程序执行 INSERT INTO 语句向表中插入想要的记录数据。

【提示】 SQL 过程可以实现与关系数据库 SQL 语句的交互，建议读者学习关系数据库 SQL 语句。

【例 8.6】 清空 Oracle 数据库表 saletest。

```
*清空关系数据库 ORACLE 中的表  saletest;
%let   jx_cnt=user=chiran password=chiran path=orcl;   /*创建连接数据库的宏变量*/
proc   sql   noprint;
connect   to   oracle (&jx_cnt);
      execute   (truncate   table   saletest) by oracle;
          disconnect   from   oracle;
quit;
```

【程序解读】

1）%let jx_cnt=user=chiran password=chiran path=orcl;：定义创建连接数据库的宏变量。

2）connect to oracle (&jx_cnt);：连接 Oracle 数据库语句。

3）execute (truncate table saletest) by oracle;：通过 execute 语句执行 truancate table 语句清空表 saletest。

4）disconnect from oracle;：断开与 Oracle 数据库的连接。

5）quit：SQL 过程的退出语句为 quit。

【提示】 本例是 SQL 过程与 Oracle 数据库连接应用的另一种方式，此过程可以执行关系数据库 SQL 的基本操作语句。

【例 8.7】 删除 Oracle 数据库表 sale_beijing 中 diqu 字段为北京的数据。

```
*根据条件删除符合条件的记录数据，关系数据库 ORACLE 中的表 sale_beijing;
%let    jx_cnt=user=chiran password=chiran path=orcl; /*创建连接数据库的宏变量*/
proc  sql  noprint;
connect   to   oracle (&jx_cnt);
     execute ( delete from sale_beijing
                          where diqu LIKE '北京%') by oracle; /*删除 diqu 字段为北京的数据*/
          disconnect   from   oracle;
quit;
```

【程序解读】

execute (delete from sale_beijing where diqu LIKE '北京%') by oracle;：删除 diqu 字段为"北京"的数据，通过 where 条件语句和 like 模式匹配的应用删除符合条件的数据。

【例 8.8】 查询 Oracle 数据库交易表 trans_jl 中日期为 2012-08-01 的交易数据信息。

该实验的具体步骤如下：

1）首先在 Oracle 数据库创建表 sale，创建表语句如下：

```
-- Create table
create   table   trans_jl
(
  empno      number(4)   not null,
  diqu       varchar2(10),
  sal         number(7,2),
  deptno     number(2),
  trans_dt date
)
Tablespace   users;
```

2）向表 sale 插入数据，脚本如下：

```
insert   into   trans_jl   values(1001,'北京市',12876.23,10,to_date('29-07-2012', 'dd-mm-yyyy'));
insert   into t rans_jl   values(1002,'上海市',26872.87,12,to_date('28-07-2012', 'dd-mm-yyyy'));
insert   into   trans_jl   values(1003,'天津市',39871.86,13,to_date('26-07-2012', 'dd-mm-yyyy'));
insert   into   trans_jl   values(1004,'山东省',55873.62,15,to_date('25-07-2012', 'dd-mm-yyyy'));
insert   into   trans_jl   values(1005,'河北省',65879.29,16,to_date('01-08-2012', 'dd-mm-yyyy'));
insert   into   trans_jl   values(1006,'广东省',95872.88,18,to_date('01-08-2012', 'dd-mm-yyyy'));
commit;
```

3）SAS 程序如下：

```
*查询 ORACLE 数据库交易表 trans_jl 中日期为 2012-08-01 的交易数据信息。
libname   jx   oracle user=chiran password=chiran path=orcl;
proc   sql;
 select *   from   jx.trans_jl
      where trans_dt=(select bksj from jx.jxfpb);/*根据 trans_dt 交易时间查询符合条件的数据，翻盘表 jxfpb 存储的是时间字段 bksj*/
quit;
```

程序执行后输出窗口显示如图 8-2 所示。

```
输出 - （无标题）
                         SAS 系统                    2012年08月

EMPNO  DIQU            SAL  DEPTNO            TRANS_DT
------------------------------------------------------------
 1005  河北省     65879.29      16    01AUG2012:00:00:00
 1006  广东省     95872.88      18    01AUG2012:00:00:00
```

图 8-2　交易时间为 2012-08-01 的数据信息

【程序解读】

where trans_dt=(select bksj from jx.jxfpb);：查询语句组合应用，where 语句中借用另一个翻盘表 jxfpb 来控制时间。此翻盘表存储的时间为 2012-08-01，每天跑批时自动更新。

【提示】 此处重点学习翻盘表 jxfpb 的应用，在业界经常应用这种处理方式，此表每天跑批时自动更新。

8.1.3　SAS装载数据到Oracle数据库

实际的业界开发中把来自外围的数据装载到大型数据库中存储，处理数据装载的过程称为 ETL 过程。一般的处理步骤如下：

① 首先对外部送过来的数据进行处理，转换成 SAS 能够处理的数据集，生成符合条件的数据，这一过程可以对外部数据文件进行过滤，转换生成符合条件的数据集。

② 对生成的 SAS 数据集通过 SAS 内部的 APPEND 过程装载到 Oracle 数据库对应的目标表中。

【例 8.9】 外部数据文件存储在“d:\jx\ jx_fgwj.dat”，请装载到 Oracle 数据库目标表 jx_append_inf 中。

```
*外部数据处理，生成 SAS 数据集;
libname   jx   oracle   user=chiran   password=chiran   path=orcl;
%let   f1 =d:\jx\jx_fgwj.dat;
%let   filjx = "&f1";
Data   appendinf ;
Infile   &filjx   dlm='|'   lrecl=389   dsd   missover   firstobs=1   obs=3 ;
input id                    :$8.
         type               :$3.
```

```
        lxh                 :$32.
        lx_nane             :$40.
        c_type              :$3.
        ex_ins              :$30.
        sf                  :$1.
        sfs                 :$1.
        vf                  :$1.
        blk                 :$1.
        sfmo                :$1.
        sfzh                :$1.
        dte_rq              :yymmdd10.
        crlimit             :20.2
        using               :20.10
        pdc                 :20.10
        pdd                 :20.10
        score_c             :20.2
        score_d             :20.2
        score_scale         :$6.
        r_increase          :$4.
        l_increase          :20.2
        s_c_per             :20.2
        s_c_tem             :20.2
        score_re_code       :$3.
        score_dt            :$8.
        h_re_f              :$1.
        h_ref_c             :$6.
        debrecord           :$20.
;

run;
*生成的 SAS 数据集 appendinf 装载到 ORACLE 数据库目标表 jx_append_inf;
proc   append   base=jx.jx_append_inf
      ( bulkload=no
        dbsastype=(
      dte_rq          ='DATE'
     crlimit         ='NUMERIC'
     using           ='NUMERIC'
     pdc             ='NUMERIC'
     pdd             ='NUMERIC'
     score_c         ='NUMERIC'
     score_d         ='NUMERIC'
     l_increase      ='NUMERIC'
     s_c_per         ='NUMERIC'
     s_c_tem         ='NUMERIC'
      )
      nullchar=NO /*告诉 SAS 系统缺失值是以 NULLCHARVAL=指定值替换*/
```

```
nullcharval=" "
)
data=appendinf;
run;
```

程序执行后查询 Oracle 数据库目标表 jx_append_inf，数据信息部分截图如图 8-3 所示。

	ID	TYPE	LXH	LX_NANE	C_TYPE	EX_INS	SF	SFS	VF	BLK	SFMO	SFZH	DTE_RQ
1	20111231	123	1234580406647698	刘小红	100	130621198345040664	1	0			0	0	2007-5-8
2	20111231	245	2236071532747168	马小明	100	110125198345153274	1	0			1	0	2011-11-15
3	20111231	567	3128081555313257	杨小华	100	120223198345155531	1	0			0	0	2007-10-25

图 8-3　jx_append_inf 表数据信息

【程序解读】

1）Infile &filjx dlm='|' lrecl=389 dsd missover firstobs=1 obs=3 ;：infile 语句读取外部数据，firstobs=1 表示从第一条记录取， obs=3 表示取到第三条记录结束。

2）proc append base=jx.jx_append_inf ：通过 APPEND 过程装载数据到目标表 jx_append_inf。

3）bulkload=no：调用 SAS 系统的 SAS/Access 模块动态逻辑库引擎装载数据到 Oracle 表中;如果 bulkload=yes，则调用 Oracle 的 SQL*LOADER 方式装载。

4）dbsastype=：若数据库中所要装载的表的变量为数值类型和日期类型，则必须通过该语句指定数据类型，通过 SAS 系统告诉 Oracle 数据库变量对应的类型。

5）nullchar=no：此语句告诉 SAS 系统缺失值是以 NULLCHARVAL=指定值替换。

6）nullcharval=" "：缺失值用空值替换。

7）data=appendinf;：指定要装载的数据集。

8.1.4　Oracle数据解数到外部数据文件

SAS 系统强大的功能还在于与数据库的交互能力，通过逻辑库就可以建立与数据库连接的桥梁。SAS 与 Oracle 数据库建立好连接通道，可以对数据库里的数据进行处理，就像处理数据集一样。对于把数据库里表数据解数到指定目录文件和处理数据集的方式一样，需要先建立 SAS 与 Oracle 数据库的连接通道，其他处理方式都一样。

具体实现步骤如下：

1）建立 SAS 与 Oracle 数据库的连接通道。

LIBNAME 逻辑库名 ORACLE USER=登录数据库用户名 PASSWORD=登录数据库密码 PATH=数据库实例;

【注意】 Oracle 中所登录的数据库叫实例，即用户登录的数据库名。

2）解数到指定目录文件和处理 DAT 解数方式一样，只是需要通过逻辑库引入数据库中的表。

【例 8.10】 Oracle 数据库表数据输出到文件，取 JX_APPEND_INF 表中 type 为 123，保留字段 id、type、lx_nane、dte_rq 和 crlimit，输出到存储路径文件“d:\jx\ appendinf123.dat”。

```
libname jx oracle user=chiran password=chiran path=orcl;
```

```
%let  jx_connect=user=chiran password=chiran path=orcl;
      /*建立与 Oracle 数据库连接的逻辑库*/
%let  lj='d:\jx\appendinf123.dat';  /*建立指向目录的文件路径的宏变量*/

proc  sql  noprint;

    create  table  tmp (
  id                    char(8),
  type                  char(3),
  lx_nane               char(40),
  dte_rq                char(8),
  crlimit               char(20)
) ;
  Connect  to  oracle (&jx_connect);
   Insert  into  tmp
   Select  * from  connection  to  oracle
        ( select
  id ,
  type ,
  lx_nane,
  to_char(dte_rq,'yyyymmdd'),
  to_char(crlimit,'9999999999999999D00')
  /*数据库中此字段 18.2，整数位占 15 位，小数点占一位，两位小数，先转换为字符，并对小数
指定位数，9 表示整数位，D00 表示小数位，两个零表示两位小数*/
   from  JX_APPEND_INF  where  type='123');

        disconnect  from  oracle;
quit;
data _null_;
     set   tmp;
     file  &lj  lrecl=84; /*file 语句把数据写入到指定目录的文件中*/
     put  @1   id         $8.    /*PUT 语句把表数据输出到外部文件*/
          @9 '|'
          @10   type     $3.
          @13 '|'
          @14   lx_nane  $40.
          @54 '|'
          @55   dte_rq   $8.
          @63 '|'
          @64   crlimit  $20.
          @84  '|'
     ;
run;
```

【程序解读】

1）Oracle 数据库的数据表中含有时间类型和数值类型的字段，需要先通过 SQL 语句处理，生成符合规范的数据，to_char(dte_rq,'yyyymmdd')把时间类型 dte_rq 字段转换为字符类型

的 yyyymmdd 格式。

2）to_char(crlimit,'999999999999999D00')：对数据库中字符为数值类型的要进行转换，数据库中 crlimit 字段 18.2，整数位占 15 位，小数点占一位，两位小数，先转换为字符，并对小数指定位数，9 表示整数位，D00 表示小数位，两个零表示两位小数。

3）数据步 data _null_;中通过 file 语句和 put 语句把数据输出到指定的文件中。

打开生成的 d:\jx\ appendinf123.dat 文件，如图 8-4 所示。

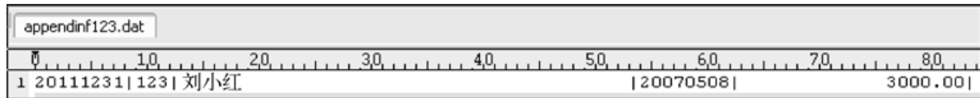


图 8-4　appendinf123.dat 解数文件显示数据信息

【提示】这里用到了 SQL 过程生成临时数据集，并对日期和数值字段通过函数进行转换。

【例 8.11】 对例 8.10 进行改造，解数文件加入表头信息“客户销售数据”，然后输出解数数据到文件 d:\jx\appendinfbt.dat。

```
libname   jx   oracle   user=chiran   password=chiran   path=orcl;
%let   jx_connect=user=chiran password=chiran path=orcl;
        /*建立与 Oracle 数据库连接的逻辑库*/
%let   lj='d:\jx\appendinfbt.dat';   /*建立指向目录的文件路径的宏变量*/

proc   sql   noprint;

    create   table   tmp (
  id                      char(8),
  type                    char(3),
  lx_nane                 char(40),
  dte_rq                  char(8),
  crlimit                 char(20)
) ;

  Connect   to   oracle (&jx_connect);

   insert   into   tmp

   select   *   from   connection   to   oracle

        ( select
  id ,
  type ,
  lx_nane,
  to_char(dte_rq,'yyyymmdd'),
  to_char(crlimit,'999999999999999D00')
  /*数据库中此字段 18.2，为 16 为整数，两位小数，先转换为字符，并对小数指定位数，9 表示
整数位，D00 表示小数位，两个零表示两位小数*/
   from   jx_append_inf );
```

```
            disconnect    from    oracle;
quit;
/*表头信息解数到文件*/
data    _null_;
    file    'd:\jx\appendinfbt.dat';
    put '客户销售数据';
run;
/*数据信息解数到文件*/
data    _null_;
        set    tmp;
        file    &lj    lrecl=84    mod;
/*mod 参数应用，追加不覆盖*/
        put    @1    id            $8.       /*PUT 语句把表数据输出到外部文件*/
               @9    '|'
               @10   type        $3.
               @13   '|'
               @14   lx_nane   $40.
               @54   '|'
               @55   dte_rq      $8.
               @63   '|'
               @64   crlimit      $20.
               @84   '|'
        ;
run;
```

【程序解读】

1）data _null_;

```
file 'd:\jx\appendinfbt.dat';
put '客户销售数据';
run;
```

数据步语句先通过 PUT 语句写表头信息到数据文件。

2）第二个数据步通过语句 file &lj lrecl=84 MOD;中加入 MOD 参数把 Oracle 数据库数据解数到指定文件，此参数只是在原来生成的数据文件后面追加数据，不覆盖原数据文件。

打开生成的 d:\jx\ appendinfbt.dat 文件，如图 8-5 所示。

```
appendinfbt.dat
0         10        20        30        40        50        60        70        80
1 客户销售数据
2 20111231|123|刘小红                                  |20070508|            3000.00|
3 20111231|245|马小明                                  |20111115|            5000.00|
4 20111231|567|杨小华                                  |20071025|            1000.00|
```

图 8-5 appendinfbt.dat 数据显示

8.1.5 条件过滤取Oracle数据库中的数据

优秀的过滤程序可以快速地从数据库中提取有效的数据，从而为分析做准备。

【例 8.12】 提取风险表 xdrish 数据中工作类型为“Other”，且好客户标志为“1”的数据，

生成数据集 goodtype。

```
%macro  fetch(v_type,v_good);
libname  jx  oracle user=chiran password=chiran path=orcl;
libname  csj  'd:\jx';  /*建立指向目录的文件路径的宏变量*/
data  csj.goodtype ;
   set  jx.xdrish;
   where  job_type="&v_type"  and  good_bad_custer=&v_good; /*查询条件*/
run;
%mend;
%fetch(Other,1); /*宏过程调用*/
```

【程序解读】

1）%macro fetch(v_type,v_good);：通过宏过程实现，定义宏过程名为 fetch，宏形参 v_type 为工作类型，v_good 为好客户标志。

2）set jx.xdrish;：读取 Oracle 数据库中的数据表 xdrish。

3）where job_type="&v_type" and good_bad_custer=&v_good;：通过 WHERE 语句实现条件过滤。And 表示且的意思，把两个条件都满足的数据取出来，生成数据集 goodtype。

4）%fetch(Other,1);：宏过程调用，传递实参 Other 和 1 分别给形参。

【例 8.13】 对例 8.12 进行改造，通过 IF 语句实现过滤。

```
%macro  fetchif(v_type,v_good);
libname  jx  oracle  user=chiran  password=chiran  path=orcl;
libname  csj  'd:\jx';  /*建立指向目录的文件路径的宏变量*/
data  csj.goodtypeif ;
  set jx.xdrish;
  if  job_type="&v_type"  and  good_bad_custer=&v_good; /*查询条件*/
run;
%mend;
%fetchif(Other,1);
```

【程序解读】

if job_type="&v_type" and good_bad_custer=&v_good; ：通过 IF 语句过滤。

【提示】 WHERE 语句与 IF 语句都能实现条件过滤，但 WHERE 语句效率高，批量提取；IF 语句是一条条提取。因此建议能用 WHERE 语句过滤就不要用 IF 语句过滤。

【例 8.14】 提取交易表 trans_jl 中日期为 2012-08-01 的数据，生成数据集 trans20120801，供其他部门分析数据时用。

```
%macro  trans;
libname  jx  oracle user=chiran password=chiran path=orcl;
proc  sql  noprint;
  select  bksj  format  50.  into :v_dte  from jx.jxsj;/*查询变量日期赋值给变量 v_indte*/
quit;
data  trans20120801;
  set  jx.trans_jl;
   where  trans_dt=&v_dte; /*查询条件*/
```

```
run;
%mend;
%trans; /*调用宏过程*/
/*查看生产的数据集*/
 proc  print  data=trans20120801;
 run;
```

查看数据集，输出窗口显示如图 8-6 所示。

输出 －（无标题）

SAS 系统　　　2012年08月09日 星

Obs	EMPNO	DIQU	SAL	DEPTNO	TRANS_DT
1	1005	河北省	65879.29	16	01AUG2012:00:00:00
2	1006	广东省	95872.88	18	01AUG2012:00:00:00

图 8-6　trans20120801 数据集显示

【程序解读】

1）SQL 过程中通过 jxsj 时间翻盘表控制时间，此时间每天更新。目前 bksj 字段时间为 20120801，通过语句 select bksj format 50. into :v_dte from jx.jxsj；赋值时间给宏变量 v_dte。

2）where trans_dt=&v_dte; ：过滤条件，trans_dt=&v_dte 值。

【例 8.15】 请从 Oracle 数据库中的数据表 xdrish 中取出第 6～260 条记录。

```
libname  jx  oracle user=chiran password=chiran path=orcl;
libname  csj  'd:\jx';  /*建立指向目录的文件路径的宏变量*/
data  csj.xdrish206;
    set  jx.xdrish (firstobs=6 obs=260);
     /* firstobs=6 控制读取行记录的开始位置，obs=260 控制读取行记录的结束位置*/
run;
```

【程序解读】

set jx.xdrish(firstobs=6 obs=260); ：firstobs=6 表示从第 6 条记录开始取，obs=260 表示取到第 260 条记录结束。

8.2　信用卡交易流水数据提取案例

随着信用卡的普及，刷信用卡消费已经成为主流交易方式，通过 SAS 系统与 Oracle 数据库的交互应用，可以提取信用卡交易流水数据分析交易信息，以便发现客户的交易规律，从而对客户采取相应的营销策略与服务推广。

【例 8.16】 从 Oracle 数据库交易表 trans_flow 中提取客户卡号为“1000000000000008”、时间为 201207-01～2012-31 期间的数据，分析客户的消费行为。数据存储在 d:\jx\tran_flow.dat。

本案例交易表 trans_flow 字段说明，如表 8-1 所示。

表 8-1　trans_flow 表字段结构说明

字　　段	说　　明
cust_id	客户号
card_num	卡号
name	姓名
jiaoyie	交易额
jiaoyi_dt	交易日期
jiaoyi_address	交易地点
jiaoyi_type	交易类型： 0：旅游消费 1：商场消费 2：养生消费 3：网购消费

具体实现步骤如下：

1）创建 Oracle 数据库中的目标表 trans_flow。

```
-- Create table
create   table   trans_flow
(
   cust_id       number(6)   not   null,
   card_num    char(18),
   name      varchar2(10),
   jiaoyie      number(7,2),
   jiaoyi_dt    date,
   jiaoyi_address    varchar2(30),
   jiaoyi_type    char(1)
)
tablespace    USERS;
```

2）创建 Oracle 数据库中的时间翻盘表 begi_end_sj。

```
Create    table    begi_end_sj
(
 begi_dt    date,
 end_dt    date
)
tablespace    users;
```

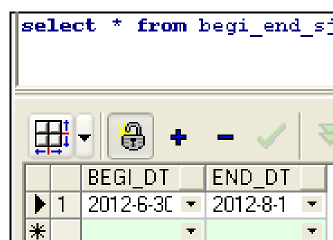


图 8-7　begi_end_sj 表信息

查询时间翻盘数据如图 8-7 所示。

3）SAS 程序实现外部数据装载到 Oracle 数据库。

```
*外部数据处理，生成 SAS 数据集;
libname   jx   oracle   user=chiran   password=chiran   path=orcl;
%let   f1 =d:\jx\tran_flow.dat;
%let   filjx = "&f1";
data transflow;
   Infile   &filjx   dlm='|'   lrecl=82   dsd   missover   ;
```

```
   input   cust_id              :6.
           card_num             :$18.
           name                 :$10.
           jiaoyie              :7.2
           jiaoyi_dt            :yymmdd10.
           jiaoyi_address       :$30.
           jiaoyi_type          :$1.
;
run;
*生成的 SAS 数据集 transflow 装载到 Oracle 数据库目标表 trans_flow;
proc   append   base=jx.trans_flow
       ( bulkload=no
          dbsastype=(
            cust_id          ='NUMERIC'
            jiaoyie             ='NUMERIC'
       jiaoyi_dt          ='DATE'
       )
       nullchar=NO /*告诉 SAS 系统缺失值是以 NULLCHARVAL=指定值替换*/
       nullcharval=" "
       )
       data=transflow;
       run;
```

4）提取 Oracle 数据库中的目标表 trans_flow 进行数据分析。

```
Libname   jx   oracle user=chiran password=chiran path=orcl;
proc sql noprint;
   select   begi_dt   format   50.   into   :v_begidt    from   jx.begi_end_sj;
/*查询变量日期赋值给变量 v_begidt*/
   select   end_dt   format   50.   into   :v_enddt   from   jx.begi_end_sj;
/*查询变量日期赋值给变量 v_enddt*/
quit;
data   trans201207;
     set   jx.trans_flow;
     where   &v_begidt<jiaoyi_dt<&v_enddt   and   card_num='100000000000000008';
      /*查询条件*/
run;
```

5）调用过程分析。

```
/*调用 means 过程分析*/
proc   means   data=trans201207 ;
var   jiaoyie;
run;
```

程序执行后输出窗口显示如图 8-8 所示。

输出 - （无标题）

SAS 系统　　2012年08月09日 星其

MEANS 过程

分析变量：JIAOYIE JIAOYIE

N	均值	标准偏差	最小值	最大值
21	1547.14	1013.45	320.1300000	3020.13

图 8-8　trans201207 数据集分析结果

【程序解读】

1）select　begi_dt　format　50.　into :v_begidt　from jx.begi_end_sj;：从数据库翻盘表 begi_end_sj 取日期 begi_dt(开始日期)赋值给变量 v_begidt。

2）select　end_dt　format　50.　into :v_enddt　from jx.begi_end_sj;：从数据库翻盘表 begi_end_sj 取日期 end_dt (结束日期)赋值给变量 v_enddt。

3）where &v_begidt<jiaoyi_dt<&v_enddt and card_num='1000000000000000008'; ：过滤条件，取日期范围内且卡号为 1000000000000000008 的数据。

4）调用 means 过程对 jiaoyie（交易额）进行分析。

【分析结果解读】

此客户月平均刷卡消费为 1 547.14 元，最高消费额为 3 020.13 元，最低消费额为 320.13 元。

第 9 章　ODS基础与综合案例

9.1　ODS基础

传统的 SAS 系统输出形式是每个过程只能自己输出结果，SAS 系统版本 7 开始支持 ODS（Output Delivery System，输出传输系统）输出各类格式文件控制。ODS 可以把过程步中产生的输出以指定的文件格式输出到目录，可以输出 PDF、HTML 等格式文件。

9.1.1　ODS概述与功能

（1）ODS 概述

SAS 运行其程序数据及其加工结果的输出中，最常用的方式是在 OUTPUT 窗口以文本的形式显示分析结果，而图形则在 GRAPH 窗口显示。这些显示方法都有一定的局限性：

1）在 OUTPUT 窗口的结果不便于转换成 SAS 数据集作进一步的分析处理。

2）在 OUTPUT 窗口是等宽的字体列表方式显示的，它的表格无法在其他字体的环境下使用。

3）每个过程提交后其输出是一个整体，难于进行挑选。

ODS 提供了输出多样化的目标格式文件，从而能够执行和控制所有 SAS 过程输出的格式。通过 ODS，可以不受限制地以各种美观的方式来报告和显示分析结果。创建的 SAS 输出可以采用多种输出格式，如 HTML、PDF、RTF、PCL，以及全新的"ODS 文档"格式。

ODS 既省时，又节约资源，还方便控制输出的结构和层次，可以生成多个 ODS 输出结果，而不必反复运行各个过程或数据查询。此外，通过重新整理、复制或删除表甚至是过程和数据查询的全部输出，可以定制或修改输出的层次结构。总之，ODS 可以控制文档内容的去留以及显示格式，而不必反复运行同一过程和数据查询。

ODS 提供了模板定义，可以定义过程和 DATA 步输出的结构。修改这些定义或使用 PROC TEMPLATE 自行创建定义，可定制输出，同时可用 TEMPLATE 过程创建和修改为输出选定或创建的任意标记集。

ODS 实现方式是通过 3 个基本组件来实现的：数据组件、表定义组件（如列的顺序和行）和输出目标文件格式（如 HTML、PDF、DAT、TXT 和 CSV）。

（2）ODS 功能

ODS 的主要功能是用来控制输出的，把数据步或过程步产生的输出结果输出到指定的文件类型格式。通过 ODS 全局语句可以控制输出结果到指定的目录和文件格式。ODS 对数据步或过程步的输出可以根据需求输出到数据集或更多文件类型格式，如 HTML、DAT、PDF、RTF 和 CSV 等文件类型的格式。对每个过程的输出结果分为一个或多个对象，使用者可选择全部或部分对象输出显示。对每个过程的输出对象，可选择不同的传送目标和显示格式，也可以挑选和剔除过程输出的某些部分，同时也可以将过程的结果输出到指定的数据集。交互操作环境的结果窗口组织和管理输出的对象。

9.1.2 ODS定义与应用

（1）ODS 输出传输各类格式文件语法定义

语法格式：ODS output-format <选项>;

SAS 程序代码;

ODS output_format close;

【语法解读】

ODS：ODS 传送输出的关键字。

output-format：指定输出的文件格式，可以取的格式有 HTML、PDF 和 RTF 等。

<选项>：ODS 全局语句中的选项参数，如 file=、contents=、frame=、gpaht=和 close 等选项。

SAS 程序代码：SAS 系统中调用内部过程对数据集进行分析处理的 SAS 程序。

ODS output_format close：指定关闭某个格式文件语句。

（2）ODS LISTING

ODS LISTING 是 SAS 系统默认的输出结果传输目标状态，默认以文本形式输出结果到 OUTPUT 窗口显示；图形输出到 GRAPH 窗口显示。

语法格式：ODS LISTING <action>;

ODS LISTING <datapanel=number|data|page> <file=file-specification>;

【语法解读】

ODS LISTING：告诉 SAS 系统采用 ODS 的 LISTING 输出方式，固定语法。

Action 选项具有以下 4 个动作：

1）关闭 LISTING 默认输出目标。

2）指定要排除的输出对象。

3）选择输出包含的输出对象。

4）对当前要排除的对象或选择的对象信息输出到 SAS 日志。

Action 具有的选项语句如表 9-1 所示。

表 9-1 Action 选项说明

选　　项	说　　明
CLOSE	关闭输出到 OUTPUT 窗口
EXLUDE exclusion(s)|ALL|NONE	指定 LISTING 要排除输出的对象
SELECT selection(s)|ALL|NONE	指定 LISTING 要选择输出的对象
SHOW	指定当前选择或排除对象信息写到日志中

datapanel=选项语句主要用来拆分数据集。每个列和行称为一个数据面板。datapanel=选项功能说明如表 9-2 所示。

表 9-2 datapanel=选项功能说明

选　　项	说　　明
number	指定一个数据面板写入的观测记录数
data	指定内存中存储面板的大小
page	默认此项，数据面板大于 200，用 datapanel=200 指定

file=选项指定输出的文件，可以取的选项分两种情况，如表 9-3 所示。

表 9-3　file=file-specification 可以取的选项说明

file-specification	说　　明
外部文件路径	指定输出到外部文件，写物理路径和文件名
文件标识	取 FILENAME 语句声明的文件标识

【提示】 如果关闭了 LISTING，可以通过执行 ODS LISTING 打开 SAS 默认输出。

实际开发中为了提高程序运行的效率，不用的管道可以通过 ODS 中的 Close 动作关闭。

（3）ODS 应用

ODS 主要用来生成各种类型的外部文件，通过 ODS 输出系统可以生成 PDF、RTF、HTML、XML、PS、DAT、TXT 等类型文件，从而把 SAS 分析系统的分析结果或数据集打印信息以文件的形式存储保存。

9.2　ODS综合案例

ODS 实际的业界应用是多个 ODS 语句组合的整体联合应用，这也体现了 ODS 输出传输系统强大的功能。

9.2.1　ODS输出PDF文件

PDF 格式文件是经常用到的一类格式文件，通过 ODS 输出传输系统可以帮助生成各种样式的 PDF 格式文件。PDF 格式文件需要通过 Adobe Acrobat Reader 来进行读取，需要有 Adobe 阅读工具。

语法格式：ODS　pdf　file=‘ 物理存储路径\file_name.pdf ’;
　　　　SAS 程序语句
　　　　ODS　pdf　close;

【语法解读】

pdf：定义输出目标为 PDF 格式文件的关键字。

file=‘物理存储路径\file_name.pdf’：定义输出 PDF 格式文件的物理存储路径和文件名，不指定物理路径系统就默认输出到默认物理路径。

ODS pdf close：close 语句关闭 PDF 格式输出。

【例 9.1】 ODS PDF 语句的应用，数据集 class1 打印输出到 PDF 格式文件，文件名为 class1_pdf。

```
*pdf 文件输出;
libname   jx   'd:\jx';
ods   pdf   file='d:\jx\class1_pdf.pdf';   /*生成 PDF 格式设置*/
proc   print   data=jx.class1;
run;
ods   pdf   close;   /*关闭 pdf 输出*/
```

【程序解读】

1）数据集 class1 存储路径文件夹为“d:\jx”，程序执行创建逻辑库语句，通过逻辑库执行数据集存储路径。

2）ODS pdf file='d:\jx\class1_pdf.pdf'：ODS 和 pdf 是关键字，pdf 关键字告诉 SAS 系统 ODS 输出信息以 pdf 格式的文件存储到 fiel=语句指定的文件 class1_pdf.pdf，存储到目录 d:\jx 文件夹下。

3）执行 print 打印过程，同时写打印信息到 class1_pdf.pdf 文件。

4）ODS pdf close：关闭 pdf 通道。

【例 9.2】 对例 9.1 进行改造，打印过程输出信息存储到“d:\jx”目录下，存储为 pdf 格式文件，文件名为 class，并关闭默认输出窗口。

```
*pdf 文件输出;
libname  jx  'd:\px';
ods  listing  close;  /*关闭默认输出到 OUTPUT 窗口*/
ods  pdf  file='d:\jx\class.pdf';  /*生成 PDF 格式设置*/
proc  print  data=jx.class1;
run;
ods  pdf  close;  /*关闭 PDF 输出*/
ods  listing;    /*打开默认输出 OUTPUT 窗口*/
```

【程序解读】

1）执行 ods listing close，关闭默认输出到 OUTPUT 窗口。

2）执行 ods pdf close，关闭 PDF 通道。

3）执行 ods listing，打开默认输出到 OUTPUT 窗口。

【例 9.3】 调查某个地区 50 个人的年收入情况，求出年收入最小值、最大值、均值和全距，分析结果输出到路径“d:\jx”文件夹下，保存为 sr.pdf'。

```
libname  jx  'd:\px'; /*定义存储逻辑库*/
ods  listing  close;  /*关闭默认输出到 OUTPUT 窗口*/
ods  pdf  file='d:\jx\sr.pdf';  /*生成 PDF 格式设置*/
%let  path= D:\jx\sr;  /*定义外部文件路径*/
%let  type=.txt;
%let  fil= "&path&type";
data  jx.sr;  /*数据集存储到指定逻辑库*/
      Infile  &fil  ;
            input          id  :$4.
                           year_money
                 ;
run;
proc  univariate  data=jx.sr  freq  normal  plot;
  var  year_money; /*指定分析变量*/
run;
ods  pdf  close;  /*关闭 pdf 输出*/
ods  listing;    /*打开默认输出 OUTPUT 窗口*/
```

【程序解读】

univariate 过程分析结果存储到 ODS　pdf 语句中 file='d:\jx\sr.pdf'指定路径文件的 sr.pdf 文件格式中。

打开此文件可以看到如图 9-1 所示的部分 PDF 格式文件显示的输出报告，分析报告以 pdf 格式保存在物理路径下。

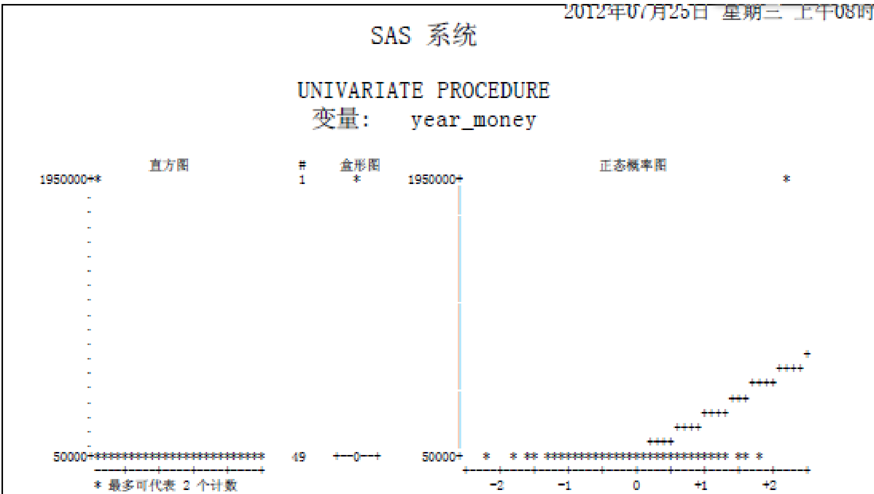


图 9-1　sr.pdf 格式文件

9.2.2　ODS输出HTML文件

HTML（Hypertext Markup Language，超文本格式文件）提供了一种输出文件的格式。

语法格式：ODS HTML file='html-file-specification' <option> <style='style-definition'>;

SAS 分析输出程序;

ODS html close;

【语法解读】

file='html-file-specification.html'：指定输出的 HTML 文件名。

<option>：通过选项对 HTML 格式文件进行设置。可以指定 4 个 HTML 文件，常用选项如表 9-4 所示。

表 9-4　option 常用选项

选　项	功 能 说 明
body='body-file-name.html'	输出的 HTML 文本体
Contents='contents-file-name.html'	包含 HTML 输出内容的表，通过超连接链接文件体
Frame='frame-file-name.html'	合并表内容、页内容和文本体内容，指定此选项时必须指定 contents=或 page=
Page='page-file-name.html'	文件体每一页的描述信息且连接文本体

Style=：选择 HTML 呈现的样式。

ODS html close：关闭 HTML 输出语句。

【提示】 SAS9 版本是 ODS HTML 输出的 HTML4.0 版本，和以前输出的 HTML3.0 版本

有相当大的差异，可以通过 ODS HTML3 声明改变成 HTML3.2 格式，也可以在 SAS 注册目录设置 HTML 版本。

【例 9.4】 对孩子的身高与父母身高的关系进行回归分析，结果以 HTML 网页的方式存储到 d:\jx 目录文件夹下。

```
ods   listing   close;
ods   results   off;
ods   html   path='d:\jx'
            body='shengao_bo.html'(title='父母与孩子身高关系')
            contents='shengao_con.html'(title='身高分析')
            frame='shengao_fram.html'(title='父母与孩子身高公式')
            newfile=proc;
*对外部数据处理;
%let   path= D:\jx\shengao;   /*定义外部文件路径*/
%let   type=.txt;
%let   fil= "&path&type";
libname   jx 'd:\jx';   /*定义逻辑库*/
data   jx.shengao;   /*数据集存储到指定逻辑库*/
      Infile   &fil dlm='|'   dsd   missover;
          input            haizi :4.
                                  father   :4.
                                  mother :4.
             ;
run;
/*调 CORR 过程分析*/
proc   corr   data=jx.shengao   nosimple;
var   father   mother   haizi;
run;
ods   html close;
ods   results off;
ods   listing;
```

【程序解读】

1）ods results off：此语句的作用是使得 proc 过程步的输出不在 results 中显示。

2）ods html path='d:\jx'

body='shengao_bo.html'(title='父母与孩子身高关系')

contents='shengao_con.html'(title='身高分析')

frame='shengao_fram.html'(title='父母与孩子身高公式')

newfile=proc;：定义 ODS 输出以 HTML 格式输出，输出的 HTML 格式文件存储到 d:\jx 文件夹下。body='语句指定主体信息存储到 d:\jx 文件夹下，以 shengao_bo.html 名字存储网页，标题名为 title=语句指定网页名称；contents=语句存储分析的过程名信息，存储到 d:\jx 文件夹下，以'shengao_con.html 名字存储网页；frame=存储过程步分析的全部信息，存储到 d:\jx 文件夹下，以 shengao_fram.html 文件名存储网页。

3）ods html close：关闭 HTML 输出。

4）ods results off：proc 过程步结束后重新打开 results。

5）ods listing：proc 过程步结束后重新打开 listing。

【例 9.5】 对例 9.4 输出到 HTML3.0 版本的 HTML 文件。

```
*html3 版本文件输出;
ods   listing   close;
ods   results   off;
ods   html3   path='d:\jx'

            body='shengao_bo3.html'(title='父母与孩子身高关系')

            contents='shengao_con3.html'(title='身高分析')

            frame='shengao_fram3.html'(title='父母与孩子身高公式')

            newfile=proc;

*对外部数据处理;
%let   path= D:\jx\shengao;   /*定义外部文件路径*/
%let   type=.txt;
%let   fil= "&path&type";
libname   jx 'd:\jx';   /*定义逻辑库*/
data   jx.shengao;   /*数据集存储到指定逻辑库*/
       Infile   &fil dlm='|'   dsd   missover;
            input            haizi :4.
                                     father   :4.
                                     mother :4.
                   ;
run;
/*调 CORR 过程分析*/
proc   corr   data=jx.shengao   nosimple;
var   father   mother   haizi;
run;
ods   html3   close;
ods   results   off;
ods   listing;
```

【程序解读】

1）ods html3：指定输出 HTML 格式为 HTML3.0 版本，其他语句相同。

2）ods html3 close：关闭 HTML3 格式输出。

9.2.3 ODS输出CSV格式文件

CSV（Comma Separated Value）格式文件是以逗号分割的一类文件，Excel 格式文件可以另存为 CSV 格式。Microsoft Office 工具通过另存为可以 CSV 格式文件，不属于 Excel 文件。两种语法格式输出文件为 CSV 格式文件，通过 csvall 输出目标输出到 CSV 格式。

语法格式 1：ODS csvall   file='物理路径\file_name.csv';

```
SAS 程序语句
ODS csvall close;
```

【语法解读】 对于生成 CSV 格式文件输出，需要通过 LISTING 通道，“file=”选项中设置文件存储路径中的文件名以扩展名 CSV 格式保存。

语法格式 2：

```
Filename 逻辑文件标识 ‘物理路径名\file_name.csv’;
ODS csvall file=逻辑文件标识;
    SAS 程序语句;
ODS csvall close;
```

【例 9.6】 将“人文与经济（Humanity and Economy）发展综合系数”指标数据输出到物理路径为“d:\jx”的文件夹下，文件格式为 CSV 格式文件。

```
*数据集存储 cvs 文件输出到指定物理路径;
data   Human_e;
  title "国民生活质量人文与经济指标分析";
 input   area   $   Human_e1-Human_e5 ;
cards;
上海  10827 23019148 11.6 15258 74
天津  10399 12938224 10.3 7979 76
北京  10377 19612368 12.6 18300 75
江苏  7779 78659903 8.2 5817 76
浙江  7524 54426891 8.3 9279 70
内蒙古  6978 24706321 7.5 3518 73
广东  6440 104303132 8.9 7482 72
辽宁  6232 43746323 7.3 4448 75
山东  6040 95793065 8.1 3919 73
福建  5900 36894216 8.3 6250 74
;
run;
option   nocenter;   /*全局变量，显示到文件的内容不居中，输出居左*/
filename   outcsv   'd:\jx\Human_e.csv';   /*定义外部输出路径和文件名逻辑标识*/
ods   listing   close;
ods   results   off;
ods   csvall   file=outcsv;   /*生成 CSV 格式设置*/
proc   print   data=Human_e noobs;
run;
ods   csvall   close;
ods   results   on;
ods   listing;   /*打开默认 OUTPUT 窗口*/
```

【程序解读】

1）option nocenter;：此语句显示到 CSV 格式文件的内容不居中，输出居左。

2）filename outcsv 'd:\jx\Human_e.csv';：定义外部输出文件路径和文件名逻辑标识。

3）ods csvall file=outcsv;：为 ODS 输出 CSV 格式设置。

4）ods csvall　close;：关闭 CSV 格式输出。

9.2.4　ODS与Oracle交互输出PDF文件

ODS 输出传输系统与 Oracle 数据库联合应用是综合应用的体现。数据步、过程步和 ODS 输出传输系统共同联合应用完成实际业务需求的任务。

【例 9.7】 提取 Oracle 数据库，根据申请得分数据，分析各类信用卡申请的信用卡数量、平均值、标准差和变异系数，分析结果以 PDF 格式存储到物理路径“d:\jx”目录下。原始数据存储在 Oracle 数据库中。

程序如下：

```
filename   exout   'd:\jx\credit_fx.pdf';   /*定义外部文件存储路径标识*/
libname   sjk   oracle   user=chiran   password=chiran   path=orcl; /*建立与数据库连接的逻辑库*/
libname   jx   'd:\jx';
ods   listing   close;
ods   pdf   file=exout;   /*生成 PDF 格式设置*/
 *数据集处理;
data   cq.creditana;/*数据集存储到指定逻辑库对应的目录中*/
        set   sjk.credit_dz;/*set 语句读取数据库中的表，建立数据集*/
run;
*对申请数据进行分析;
proc   means   data=cq.creditana   n   mean   std   cv   maxdec=3;
/*调用 means 统计过程，maxdec=2 指定输出中最多有两位小数*/
class   credit_type; /*按 credit_type 分类*/
var   cre_xlsc   cre_nxsc   cre_hysc;/*分析学历得分，工作年限得分，行业得分*/
run;
ods   pdf   close; /*关闭 PDF 输出*/
ods   listing;/*打开默认输出*/
```

【程序解读】

1）执行 filename 语句，生成外部文件标识 exout，文件标识对应的物理路径为“d:\dzwj”。

2）建立 SAS 连接 Oracle 数据库的逻辑库。

3）建立存储数据集到指定物理路径的逻辑库，执行“libname　cq　'd:\dzwj'”语句。

4）执行“ODS listing close”语句，暂时关闭不需要的输出通道。

5）执行“ODS pdf file=exout”语句，打开 PDF 输出通道，并通过 file=exout，把输出写到文件标识对应的物理路径目录中。

6）执行数据步，读取数据库中的表 credit_dz 创建数据集，存储到 cq 逻辑库指向的物理路径。

7）执行过程步，调用均值 means 过程对数据步生成的数据集进行分析，输出分析结果到 PDF 通道，存储到指定物理路径。

8）分析完成后，执行“ODS pdf close”语句，关闭 PDF 输出通道。

9）执行“ODS listing”语句，打开 SAS 默认输出通道。

9.2.5 ODS 输出TXT格式文件

TXT 格式是一种文本文档，是微软在操作系统上附带的一种文本格式，ODS 输出传输系统有两种书写语法方式生成 TXT 格式文件。

语法格式 1：ODS listing file='物理路径\file_name.txt';

```
        SAS 程序语句
      ODS listing;
```

【语法解读】

ODS listing file=：指定 LISTING 通道，“file=”选项中设置文件存储路径中的文件名以 TXT 格式保存。

通过 filename 指定外部文件存储路径生成 TXT 格式文件。

语法格式 2：

```
    Filename  逻辑文件名 '物理路径名\file_name.txt';
ODS listing file=逻辑文件名;
    SAS 程序语句;
ODS listing;
```

【提示】 通过 LISTING 管道，上面两种语法格式都能输出 TXT 格式文件。

【例 9.8】 人文与经济指标数据集输出到物理路径为“d:\jx”的目录下，存储为 TXT 格式文件，文件名为 humantxt。

```
*数据集存储 TXT 文件输出到指定物理路径;
data   Human_e;
  title "国民生活质量人文与经济指标分析";
  input   area   $   Human_e1-Human_e5 ;
cards;
上海  10827 23019148 11.6 15258 74
天津  10399 12938224 10.3 7979 76
北京  10377 19612368 12.6 18300 75
江苏  7779 78659903 8.2 5817 76
浙江  7524 54426891 8.3 9279 70
内蒙古  6978 24706321 7.5 3518 73
广东  6440 104303132 8.9 7482 72
辽宁  6232 43746323 7.3 4448 75
山东  6040 95793065 8.1 3919 73
福建  5900 36894216 8.3 6250 74
;
run;
option   nocenter;   /*全局变量，使显示到文件的内容不居中，输出居左*/
filename   outtxt   'd:\jx\humantxt.txt';
/*定义外部输出路径和文件名逻辑名，逻辑名长度不能超过命名规则长度*/
ods   listing   file=outtxt;   /*生成 TXT 格式设置，file=逻辑名*/
proc   print   data=Human_e noobs;
run;
```

```
ods    listing; /*打开默认 OUTPUT 窗口*/
```

【程序解读】

1）程序首先处理 cards 语句数据块读入的数据，生成数据集 Human_e，存储到当前临时逻辑库中。

2）filename　outtxt　'd:\jx\humantxt.txt';：建立外部存储路径文件标识和文件名，扩展名为 TXT。

3）ODS　listing　file=outtxt;：生成 TXT 格式设置，file=逻辑名指定写入的外部文件名，即 filename 语句定义的逻辑名 outxtx。

9.2.6　ODS输出DAT格式文件

DAT 格式文件并不是一种标准文件，许多软件都使用这个扩展名，但文件含义不同。例如，VCD 光盘中的 DAT 文件就可以用一般的视频播放器打开，而 QQ 的 DAT 文件中则存储了用户信息，是无法使用常规方式打开的，只有 QQ 程序可以访问。还有一些其他程序都有自己对 DAT 文件的定义，要通过其特殊的程序来打开与之相关联的 DAT 文件。本节 ODS 输出的 DAT 格式文件属于一种文本文件的扩展名，扩展名是.DAT。这种文本是纯文本，没有数据属性结构方面的信息，可以用记事本等文本工具打开。

语法格式 1：ODS listing file='物理路径\file_name.dat';

```
            SAS 程序语句
        ODS listing;
```

【语法解读】

ODS listing file=：指定 LISTING 通道，“file=”选项中设置文件存储路径中的文件名以 DAT 格式保存。

通过 filename 指定外部文件存储路径生成 DAT 格式文件。

语法格式 2：

```
    Filename  逻辑文件标识 '物理路径名\file_name.dat';
ODS listing file=逻辑文件标识;
    SAS 程序语句;
ODS listing;
```

【提示】 通过 LISTING 管道，上面两种语法格式都能输出 DAT 格式文件。

【例 9.9】 对例 9.8 进行改造，生成为 DAT 格式文件，文件名为 humandat。

```
*数据集存储 dat 文件输出到指定物理路径;
data    Human_e;
  title "国民生活质量人文与经济指标分析";
    input    area    $    Human_e1-Human_e5 ;
cards;
上海  10827 23019148 11.6 15258 74
天津  10399 12938224 10.3 7979 76
北京  10377 19612368 12.6 18300 75
江苏  7779 78659903 8.2 5817 76
```

```
浙江 7524 54426891 8.3 9279 70
内蒙古 6978 24706321 7.5 3518 73
广东 6440 104303132 8.9 7482 72
辽宁 6232 43746323 7.3 4448 75
山东 6040 95793065 8.1 3919 73
福建 5900 36894216 8.3 6250 74
;
run;
option    nocenter;    /*全局变量，使显示到文件的内容不居中，输出居左*/
filename    outdat    'd:\jx\humandat.dat';
/*定义外部输出路径和文件名逻辑名，逻辑名长度不能超过命名规则长度*/
ods    listing    file=outdat;    /*生成 DAT 格式设置，file=逻辑名*/
proc    print    data=Human_e noobs;
run;
ods    listing; /*打开默认 OUTPUT 窗口*/
```

【程序解读】

1）程序首先处理 cards 语句数据块读入的数据，生成数据集 Human_e，存储到当前临时逻辑库中。

2）filename　outdat　'd:\jx\humandat.dat';：建立外部存储路径文件标识和文件名，扩展名为.DAT。

3）ods　listing　file=outdat;：生成 DAT 格式设置，file=逻辑名指定写入的外部文件名，即 filename 语句定义的逻辑名 outdat。

9.2.7　ODS输出RTF格式文件

RTF 格式文件是许多软件都能够识别的文件格式，如 Word、WPS Office、Excel 等都可以打开 RTF 格式的文件。

RTF（Rich Text Format，多文本格式）是一种类似 DOC 格式（Word 文档）的文件，有很好的兼容性，使用 Windows“附件”中的“写字板”就能打开并进行编辑，使用“写字板”打开一个 RTF 格式文件时，将看到文件的内容。如果要查看 RTF 格式文件的源代码，只要使用“记事本”将它打开就行。可以像编辑 HTML 文件一样，使用“记事本”来编辑 RTF 格式文件。

对普通用户而言，RTF 格式是一个很好的文件格式转换工具，用于在不同应用程序之间进行格式化文本文档的传送。通用兼容性是 RTF 的最大优点。

```
语法格式：ODS    rtf    file='物理存储路径\file_name.rtf' <style='指定输出样式'>;
              SAS 程序语句
          ODS    rtf    close;
```

【语法解读】

1）RTF：定义输出目标为 RTF 格式文件的关键字。

2）file=‘物理存储路径\file_name.rtf ’：定义输出RTF格式文件的物理存储路径和文件名，不指定物理路径系统就默认输出到默认物理路径。

3）ODS rtf close：close 语句关闭 RTF 格式通道输出。

【例 9.10】 对例 9.8 进行改造，生成为 RTF 格式文件，文件名为 humanrtf。

```
*存储 rtf 文件输出到物理路径;
*数据集存储 dat 文件输出到指定物理路径;
data    Human_e;
   title "国民生活质量人文与经济指标分析";
   input    area    $    Human_e1-Human_e5 ;
cards;
上海  10827 23019148 11.6 15258 74
天津  10399 12938224 10.3 7979 76
北京  10377 19612368 12.6 18300 75
江苏  7779 78659903 8.2 5817 76
浙江  7524 54426891 8.3 9279 70
内蒙古  6978 24706321 7.5 3518 73
广东  6440 104303132 8.9 7482 72
辽宁  6232 43746323 7.3 4448 75
山东  6040 95793065 8.1 3919 73
福建  5900 36894216 8.3 6250 74
;
run;
option    nocenter;    /*全局变量，使显示到文件的内容不居中，输出居左*/
ods    listing    close;    /*关闭默认输出到 OUTPUT 窗口*/
ods    rtf    file='d:\jx\humanrtf.rtf' ;    /*生成 RTF 格式设置*/
proc      print    data=Human_e noobs;
run;
ods    rtf    close;    /*关闭 RTF 输出*/
ods    listing;      /*打开默认输出 OUTPUT 窗口*/
```

【程序解读】

1）ODS　rtf　file='d:\jx\humanrtf.rtf ' ;：打开RTF通道，把输出信息写入到RTF格式文件humanrtf.rtf'中。

2）ODS　rtf　close;：关闭 RTF 输出通道。

9.2.8　ODS 输出到打印机

ODS 输出传输系统可以输出到打印机，通过 ODS 输出传输系统语句中的 ODS PIRNTER 语句把输出对象输出到打印机打印或输出到打印文件。

语法格式：ODS printer <动作>;

　　　　　ODS printer <选项> <样式>;

【语法解读】

ODS PRINTER：SAS 系统采用 ODS 的 PRINTER 输出方式，固定语法。

Action 选项具有以下 4 个动作：

1）关闭 PRINTER 默认输出目标。

2）指定要排除的输出对象。

3）选择输出包含的输出对象。

4）对当前要排除的对象或选择的对象信息输出到 SAS 日志。

<动作>具有的选项语句如表 9-5 所示。

表 9-5　<动作>项说明

选　　项	说　　明
CLOSE	关闭打印文件输出 PRINTER
EXLUDE exclusion(s)|ALL|NONE	指定 PRINTER 要排除输出的对象
SELECT selection(s)|ALL|NONE	指定 PRINTER 要选择输出的对象
SHOW	指定当前选择或排除的对象信息写到日志中

<选项>常用选项语句如表 9-6 所示。

表 9-6　option 常用选项说明

<选项>常用项	说　　明
BACKGROUND=	指定打印文本是否要背景颜色，默认 YES。取 NO 不要背景颜色
COLOR=	对输出指定颜色模式，默认值 YES，用样式提供的颜色；COLOR=FULL 指定对文本和图形用 full 颜色；COLOR=GRAY 指定对文本和图形用灰色颜色输出；COLOR=MONO 指定对文本和图形单色输出；COLOR=NO 指定不用样式定义提供的所有颜色
FILE=	指定输出打印对象生成打印格式的文件，FILE=‘文件存储物理地址’；FILE=‘文件标识’，需要和 FILENAME 语句结合，取 FILENAME 语句定义的文件标识。默认 PCL 格式文件
PCL	指定生成的 PCL 格式文件
PDF	指定生成的 PDF 格式文件
PS	指定生成 PS 格式文件
PRINTER=	指定输出到指定打印机打印，写打印机名字
SAS	指定用 SAS 系统提供的打印模式

Style=：指定打印样式，如 D3D、BRICK、BEIGE 等。

【例 9.11】 表 9-7 是全国 9 省市居民 2011 年支出情况数据汇总资料，主要涉及生活消费支出情况的 8 个指标，进行聚类分析，并把分析结果输出到打印机。

表 9-7　全国 9 省市居民 2011 年支出情况数据

地　　区	食 品 消 费	居　　住	医 疗 花 费	交通和通信	教　　育
天津	1 117.72	1 200.16	6 000.10	800.32	6 800.87
北京	2 300.12	1 600.88	7 898.92	1 300.89	12 000.56
吉林	1 020.00	780.08	5 456.21	678.21	4 000.32
上海	2 287.15	1 889.23	8 356.21	1 500.23	15 000.21
江苏	1 317.88	467.62	163.16	293.07	6 700.21
浙江	1 838.57	798.88	326.12	496.86	8 900.96
福建	1 408.54	430.14	136.40	306.06	7 680.09
山东	1 100.13	560.97	1 678.85	221.93	298.23
广东	1 681.68	1 700.21	8 700.19	900.23	12 000.26

```
data    diaocha;
input   diqu   $   shipin   house   yiliao   jiaotong   education;
cards;
天津  1117.72 1200.16 6000.10 800.32 6800.87
北京  2300.12 1600.88 7898.92 1300.89 12000.56
吉林  1020.00 780.08 5456.21 678.21 4000.32
上海  2287.15 1889.23 8356.21 1500.23 15000.21
江苏  1317.88 467.62 163.16 293.07 6700.21
浙江  1838.57 798.88 326.12 496.86 8900.96
福建  1408.54 430.14 136.40 306.06 7680.09
山东  1100.13 560.97 1678.85 221.93 298.23
广东  1681.68 1700.21 8700.19 900.23 12000.26
;
run;

*ods   printer 输出到打印机或打印文件;
ods   listing close;   /*关闭 LISTING 通道*/
ods   printer;   /*输出到打印机*/
proc   cluster data =diaocha standard method =ward
       outtree =jltree   pseudo;
       copy   diqu;
run;
proc   tree data =jltree   horizontal;
   id diqu;
run;
ods   printer   close;   /*关闭打印管道*/
ods   listing;
```

【程序解读】

1）ods listing close;：关闭不用的 LISTING 通道，以减小运行输出压力。

2）ods printer;：把下面的分析输出到打印机。

3）ods printer close;：当打印输出完成时把打印机通道关闭。

第 3 篇　综合实战篇

第 10 章　信用卡管理系统案例

10.1　业务需求分析与架构设计流程

1．业务需求分析

系统需求分析是确定业务主题的依据。对以客户为中心的信用卡管理系统业务需求分析以及数据库业务主题的确定是进行信用卡管理系统开发的第一步。

项目需求分析可以说占了整个项目开发时间的 50%，也是所有项目开发阶段最重要、优先级别最高的阶段。需求分析确定了项目具体实现的内容，也是开展下一步工作的指南。通过一个项目里程碑基线可以看到整个项目开发阶段时刻需要回顾项目需求分析所需要实现的功能点和整体架构。

信用卡业务是银行业务的一个重要组成部分。信用卡业务中，利润率和风险控制是两个关键指标，而建立以客户为中心的管理信息系统，通过对大量信息的分析找出客户消费的行为、规律和违约特征，进而预测客户的个性化需求、及时响应客户的需求、设计出更加符合客户需要的产品和服务，同时防范信用卡风险，既可以赢得客户认可又可以提高利润率。目前大多数信用卡业务缺乏真正面向市场、面向客户的决策支持系统，对客户的判别也只停留在静止、片面、主观的水平，不能对客户做出动态的、全面的、客观的评价。为了更好地细分市场、细分客户，有针对性地进行客户营销、提高业务管理水平，建立基于数据库、联机分析处理、数据挖掘的信用卡决策支持系统，成为银行信用卡提高综合竞争力的必然选择。

信用卡是持卡人可凭卡在特约商户购物消费，或在银行提取现金，具有消费信贷功能的信用凭证。信用卡是传统金融业务与现代信息技术相结合的产物，日益成为当今银行业新兴的高速增长支付消费工具。随着信息化技术的发展，伴随着信用卡业务在我国的高速发展，信用卡风险也层出不穷，信用卡也正在成为我国金融机构不良贷款的新增来源。为此，如何有效管理和破解风险，已成为发卡行关注的重点。同时信用卡业务对外资银行开放，信用卡市场面临国内和国外的竞争，同时也面临巨大的风险控制挑战。目前信用卡业务在国内得大了很大的发展，中国的商业银行开展信用卡业务已有多年，相关数据积累相对完备。信用卡业务的经营运作也已从简单的扩大规模和信用卡发卡量到信用卡风险控制阶段，各商业银行不断推出新的服务品种和花样繁多的增值服务，提高市场占有率并强化品牌意识以获得利润。

信用卡业务不依靠于分支机构网点的特点将使银行信用卡业务面临更加严酷的竞争。信用卡业务竞争本质上就是客户的竞争，而且是优质客户的竞争。日常运作的信用卡客户交易的日常交易信息和客户服务基础设施，无法提供众多分析和数据挖掘对信用卡信息的分析和

客户多角度的分析。面对积累的大量历史数据，联机事物处理（OLTP）不能满足多维的分析和数据挖掘，数据仓库为信用卡未来的发展提供前端众多分析和数据挖掘的数据支撑，同时能应对突发的、复杂的决策分析。

当前银行业面临的问题是收益与风险同时并存，伪信息对信用卡行业造成的损失是巨大的，欺诈的手段多样化，贷款违约客户的增多，都给银行业带来了巨大的损失。为防范风险，银行业必须面对和解决当前风险控制问题，尽最大努力减少损失是所有银行面临的问题，信用卡（Credit Card）在国内的应用越来越广泛，可以说现在已经进入了一个信用卡刷卡消费的时代，办卡的条件和限制也是比较宽松的。国内信用卡面临的主要风险是伪信息办卡、信用卡套现、信用卡交易诈骗等。

目前，信用卡业务发展主要呈现以下特点：

1）信用卡市场规模逐步扩大。信用卡业务已经由消费者的潜在需求逐渐转向了快速拓展阶段，特别是随着消费者群体的年轻化，对信用卡的使用已经越来越多样化。

2）信用卡行业竞争激烈。随着信用卡发卡机构的不断增加，而且发卡机构发行的信用卡在提取现金、消费信贷等方面基本功能都比较相似，因此市场竞争也越来越激烈。同时，我国的信用卡市场正在面临着来自诸多外资银行的竞争，外资银行已经开始采取间接地与中资银行合作的方式渗入中国市场，通过其在信用卡运行、结算、配套服务等方面的优越性参与竞争，更加大了信用卡市场的竞争压力。

信用卡凭借其灵活、方便、快捷的特点，逐步渗透到消费者的日常生活中，已经成为了消费信贷的重要形式。随着信用卡的不断普及，信用卡相关风险发生的频率也越来越高，造成的损失也越来越大，给金融业带来无法预计的损失。目前，信用卡业务面临的风险主要有以下几个方面：

1）信用风险。信用卡的信用风险是指由于持卡人违反约定，不能按时足额归还透支款项，给发卡银行带来损失的可能性。信用风险是信用卡风险中最主要的风险。信用风险产生的原因主要有持卡人客观上丧失偿还信用透支的能力；持卡人的授信额度越大，信用风险也就越大；持卡人主观上没有偿还债务的意愿，持卡人故意逃避责任，拒不履行透支还款义务。

2）操作风险。操作风险是指由不完善或有问题的内部程序、人员及系统或外部事件所造成损失的风险。一些发卡行片面追求发卡量和市场占有率，放宽了信用卡申请人的审核，信用卡目标客户不断扩大，逐步向大众人群渗透。发卡机构通常采用“免担保”的发卡方式，以吸引潜在持卡人，这很大程度上依赖于持卡人的个人信用来保证信用卡业务的安全性。

3）欺诈风险。欺诈风险是指由于遭人冒名申请、伪造、盗领、失窃等原因而发生损失的可能性。

建立以客户为中心的信用卡业务数据仓库，便于方便企业各级工作人员获取各类信息，实现对客户治理、客户消费行为、成本收益、风险控制、绩效评估、营销战略等决策目标的支持，并达到风险治理和控制、客户关系治理与个性化服务、商户分析与市场策略、费用控制与利润分析四大应用目标。

2．架构设计流程

数据仓库技术是实现将数据转换为信息和知识并有效支持决策分析的重要手段。数据

架构设计作为数据仓库技术中的一项重要技术，是对系统逻辑体现结构和建模方式的描述，直接决定了数据仓库系统的可管理性和可扩展性，在数据仓库系统的建设过程中处于重要地位。

对于一个大型数据仓库项目来说，一般将设计分为 ODS、DW、DM 三大层次。设计时需要考虑以下两个方面：

（1）前端展现

前端展现是给客户的最终用户看的，客户只关心能给他们带来什么，是否能满足他们的报表、数据挖掘、查询和分析需求。

（2）架构设计中包括的内容

架构中重点是描述系统的结构，以及他们之间的关联、交互接口。BI 系统可以划分成业务模型、元数据、数据质量、接口平台、报表集市、指标库等。

业务模型是存放业务数据的结构，可以再细分成 ODS、DW（还可以细分）、DM 等。它是支撑业务分析需求的，如报表、数据挖掘、OLAP、专题应用等。而元数据为整个系统数据的形态和数据流动的过程起到支撑作用，也就是说数据从源头开始到最终的用户眼前，其来龙去脉，每个环节的状态都需要掌握。而数据质量模块是为衡量数据源质量，为 ETL 过程处理质量提供支撑。接口平台是处于源系统和数据仓库系统之间的，方便明确界定双方职责的模块。数据挖掘为数据挖掘工具提供数据支撑，报表集市为报表应用提供支持，指标库为绩效管理需求提供支持，后三者其实还可以归入业务模型一类，因为它们都是服务于分析需求的。之所以分成若干模块，是为了让架构清晰，降低这些模块之间的耦合。将系统的用户分为两大类角色：

1）系统运营角色，对系统的正常运行、维护负责。

2）业务分析角色，需要从这个系统得到数据分析的功能。

系统运营角色不直接和业务模型这个模块打交道。 业务分析角色分析数据来源都将来自业务模型模块。而在架构设计中，重点应该放在如何满足系统管理用户的需求上面。

数据仓库设计阶段，开发设计者是其架构最重要的用户；对于开发实施人员，需要进行系统部署、ETL 的开发调试、质量的稽核；设计人员需要进行模型的变更、系统调优、作系统一致性分析等；而系统管理员则需要监控 ETL 过程、监控系统运行、响应系统警报、接口数据管理等。这些都可以看做是用例。

信用卡客户管理数据仓库系统的主要功能是利用业务系统所积累的有关信用卡客户的各类数据来获取信息，从各系统提取数据装载到目标数据仓库中，为前端应用层提供联机分析的数据，满足不同的分析需求。

以客户为中心的客户管理数据仓库系统的架构主要由 3 个部分组成：建模系统、ETL 系统和 OLAP 系统。其技术实现需要考虑以下 3 个方面：

（1）建模系统

建模系统主要用于辅助设计人员进行数据仓库设计。由于模型是对现实事物的反映和抽象，所以它可以帮助设计人员更加清晰地了解客观世界。对于数据仓库而言，数据仓库建模在业务需求分析之后开始，是数据仓库构造工作正式开始的第一步，正确而完备的数据模型是用户业务需求的体现，是数据仓库项目最重要的关键技术因素。

客观管理数据仓库系统属于银行信用卡业务，其业务复杂、机构复杂、系统庞大，因此

数据仓库建模阶段必须考虑以下几个方面：

1）满足不同用户的需求。信用卡业务流程比较复杂，数据仓库系统涉及的业务用户众多，在进行数据模型设计的时候必须考虑到不同业务产品、不同业务部门、不同层次、不同级别用户的信息需求。数据仓库应该支持企业的各种业务，如信用卡客户基本信息管理业务、信用卡积分业务、信用卡申请业务、信用卡风险控制业务等不同业务的特点；不同的业务部门对信息的需求各有不同，应考虑业务、风险、催收、管理、财务等各个部门的需要；不同层次的组织所关心的信息不同，数据模型应支持各层次信息需求。

2）兼顾效率与数据粒度的需要。数据粒度和查询效率从来都是矛盾的，细小的数据粒度可以保证信息访问的灵活性，但同时却降低了查询的效率并占用大量的存储空间，数据模型的设计必须在这矛盾的两者中取得平衡。数据模型设计既可以提供足够详细的数据支持又能够保证查询的效率。

3）支持需求的变化。信用卡用户的信息需求随着市场的变化而变化，随着竞争的激化，需求变化会越来越频繁。数据模型的设计必须考虑如何适应和满足需求的变化。

4）避免对业务运营系统造成影响。信用卡客户管理数据仓库系统是一个每天都在长大的数据仓库，其信息存储每天都在增长，其运行占用很多的资源，比如网络资源、系统资源、存储资源等在进行数据模型设计的时候也需要考虑如何减少对业务系统性能的影响。

5）考虑未来的可扩展性。信用卡客户管理数据仓库系统是一个与信用卡行业同步发展的系统，数据模型作为数据仓库的灵魂必须提供可扩展的能力，在进行数据模型设计时必须考虑未来的发展，未来需要改动时不需要对数据仓库中原有的系统进行大规模的修改。

由于信用卡客户管理数据仓库的数据量比较大，所以数据模型的技术功能必须考虑周全，从技术功能划分，需要考虑以下几点：

1）分段存储区。由于数据仓库中的数据结构和组织方式具有很大的差异所以原始业务系统的数据必须经过严格的抽取、映射和转换，数据的整合过程十分复杂，通常会耗费比较长的处理时间。如果从联机系统直接抽取数据到数据仓库，则必定会占用许多系统的资源和时间，为了避免影响联机系统的运行，数据模型的设计中引入分段存储区的概念。

分段存储区是指为了保证数据移动的顺利进行而开设的阶段性数据存储空间，它是系统原始数据进入数据仓库前的缓存区。需要进入数据仓库的各个系统的数据首先直接快速传输到分段存储区，再从分段存储区经过清洗、转换、映射等复杂的数据移动处理转移到目标数据仓库中。从各系统到分段存储区的数据传输，应尽量避免进行复杂的数据处理，以保证数据的快速导入而尽量减小对各系统造成的压力。分段存储区的数据有关系数据库和文件两种不同存储方式，分别对应于不同运营系统的数据源。数据成功导入数据仓库之后，应清空分段存储区中的数据。通过分段存储区可以减少对各系统资源的占用，避免复杂数据转换对各系统的影响。实际开发运营中的经验可以看到跨网络的数据库操作会大大降低数据处理的效率，而且处理的复杂程度越高，网络对处理效率的影响越严重，分段存储区可以大大加速数据仓库后台数据数据处理过程的实现，其作为数据缓存区，可以在一定程度上屏蔽各系统变化对数据移动整合系统的影响。如果数据处理过程中发生系统故障，作为数据仓库系统的备份数据，可以直接从分段存储区进行数据仓库数据恢复，而不必再从各系统原始数据开始。

2）基础数据仓库。信用卡客户管理数据仓库的基础数据仓库存储所有最详细的业务数据。该层数据直接来源于对分段存储区数据的清洗和加工，属于未经汇总的数据，但数据的组织方式可能已经完全不同于原始的业务系统。根据业务需求的不同，基础数据仓库的组织形式以三范式模型为主，在有的系统中也可能采用星形或雪花模型。在银行业的数据仓库系统中，基础数据仓库数据包括未经汇总的客户交易数据、客户积分数据、客户账户数据、客户卡数据等基本数据。由于基础数据仓库数据是对原始业务数据的原形再现，所以数据量会非常庞大，根据不同业务的需要数据保留的时间在半年到三年不等。

3）数据集市。根据业务需求将信用卡客户管理数据仓库数据分类成几个不同的数据集市，每个数据集市完成不同的分析和查询需求，数据集市中的数据通常由基础数据仓库的详细数据聚合而来，根据数据聚合程度的不同包含轻度聚合、中度聚合和高度聚合 3 种不同的层次。汇总的方式将依据数据量的大小和使用频度综合考虑。

（2）ETL 设计

ETL 过程是数据仓库的核心。ETL 工具的选择和效率直接影响到数据仓库的效率和前段工具的应用。

信用卡系统的数据来源于前端事务环境中的联机事务数据和主机文件数据，由于事务数据存放在服务器上的数据库（如 Oracle）系统中，主机文件以文件的形式存储，因此事务数据和主机文件数据必须先通过数据抽取、转换、清晰、集成后，再装载到数据仓库中，形成符合要求的数据。不同级别的综合数据按照不同汇总粒度计算获得，并且要加上时间戳。

ETL 过程就是实现数据清洗、抽取、转换、加载的过程，属于动态过程，是数据仓库工程的一个必需的步骤。这里需要用到 ETL 工具（Extract Transformation Loading），如 SAS 工具、DataStage 工具，主要负责将信用卡客户管理数据仓库所需要的数据转换成合适的数据格式与数据内容，并加载到数据仓库中。ETL 分为 ETL 设计模块、ETL 调度模块、ETL 关系数据库抽取/转换模块、ETL 文件处理模块、ETL 加载模块、ETL 中间层管理模块。数据的抽取、转化和加载的主要过程如下：

1）ETL 设计模块定义 ETL 转换加载过程，同时生成数据源访问元数据文件、脚本元数据文件、日志元数据文件、数据库模式文件。

2）ETL 调度是执行 ETL 处理各模块的调度过程，通过调度器打开数据源访问元数据文件、脚本元数据文件、日志元数据文件，然后开启调度服务，自动将任务分配到与调度器连接的各个执行模块上执行。

3）ETL 各作业设计好后通过专门的调度工具可以调度每一个作业，根据时间设置定时启动调度作业。

（3）OLAP 系统

以客户为中心的客户管理数据仓库的目的是为 OLAP 系统服务提供数据支撑。数据仓库存储数据，主要为决策支持和数据挖掘提供服务，主要包括的数据有 3 部分：第一部分是完整的历史数据； 第二部分是已物化的数据立方体，由于物化的数据立方体已存储在数据仓库中，为挖掘节省了时间；第三部分是元数据。

信用卡客户管理数据仓库 ETL 处理采用 SAS 或 DataStage 工具实现，系统组成如图 10-1 所示。

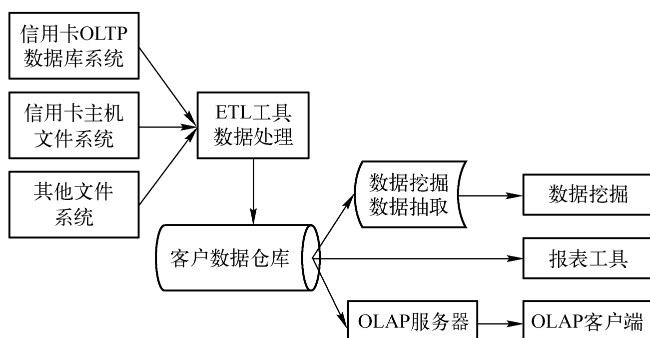


图 10-1　客户管理数据仓库系统

10.2　ETL层数据处理

ETL 是数据仓库的核心。理解 ETL 流程与设计过程是实现 ETL 的一个关键环节，也是实现数据仓库开发的核心部分。

1．客户管理系统ETL概述

以客户为中心的信用卡管理系统数据仓库的数据分别来自 OLTP（联机事物处理）、外部数据文件、各分行，因此其数据源比较复杂。实际的开发中分析外部数据源，确定提取数据的范围占用整个 ETL 开发过程的 30%的时间。

对数据仓库而言其数据的装载是在 ETL 过程这步完成的，因此合理的规划和设计是整个数据仓库中重要的环节。对于本案例而言外围数据源如下：

（1）客户信息数据文件

客户信息数据文件存储客户的基本信息，如客户姓名、性别、身份证号、原工作单位和联系方式等基本信息。

（2）账户信息数据文件

账户信息数据文件存储客户账户信息，如账户状态、账号、账户标识等基本信息。

（3）卡信息数据文件

卡信息数据文件存储卡片信息，如卡号、卡标识、客户姓名、最高额度等基本信息。

（4）交易信息数据文件

交易信息数据文件记录客户刷信用卡交易的基本信息，如交易的卡号、账号、客户姓名、交易时间、交易额度、消费方式等基本交易信息。

（5）客户消费行为数据文件

客户消费行为数据文件是对客户消费行为记录的基本信息，如卡号、客户姓名、消费额度、消费时间、消费方式、客户状态等基本消费信息。

（6）催收信息数据文件

催收信息数据文件是银行对违约客户进行记录的数据文件，记录客户违约的信息和银行催款信息，如客户姓名、违约额度、违约天数、催收员编号、催收额度、催收时间、催收方式、实际催收还款额等基本催收信息。

（7）客户积分数据文件

客户积分数据文件是银行对刷卡客户给予积分的一种消费刺激策略，记录客户刷卡给予

记入积分的信息，如客户姓名、卡号、交易额度、本月积分、总积分、兑换积分、赠送积分等基本积分信息。

（8）信用卡申请数据文件

信用卡申请数据文件是对申请信用卡客户的基本信息的采集，通过此信息来了解客户的信用度，记录申请客户的基本情况，如客户姓名、性别、工作年限、年收入、婚姻状况等基本信息。

（9）风险信息数据文件

风险信息数据文件是对风险客户情况的信息记录，通过此数据可以判定哪些客户可能违约，记录客户风险信息的数据，如客户姓名、卡号、违约类型、欠款额和违约情况等基本信息。

（10）营销信息数据文件

营销信息数据是银行和商户达成协议，刺激消费，达到共赢的一种方式，记录营销信息的数据，如营销商户、营销产品、营销评价等基本营销情况。

2．信息管理系统表结构与说明

（1）客户信息表

客户信息表存储信用卡客户的基本信息，是对客户基本资料的存储。实际应用中客户信息表数据装载一般来自 OLTP 或主机文件。客户信息表如表 10-1 所示。

表 10-1　客户信息表（cust_info）

字　段　名	英文字段名	类　　型	备　　注
客户号	Custer_id	char(8)	客户标识号
卡号	Card_id	Char(16)	
姓名	Name	Varchar2(10)	客户姓名
性别	sex	Char(1)	F：女，M：男
证件号	id	Varchar2(18)	存储证件号码
出生日期	birth	date	
婚姻状况	Marital_status	Char(1)	N：未婚，Y：已婚
学历	xl	Varchar2(10)	只填最高学历
职务	zw	Varchar2(12)	
工作年限	Work_year	Number(2)	
地址 1	addr_1	Varchar2(30)	目前所用地址
地址 2	addr_2	Varchar2(30)	身份证地址
邮编	zone	Varchar2(10)	
手机	mobile	Char(11)	
固定电话	phone	Varchar2(12)	
电子邮件	E_mail	Varchar2(30)	电子邮箱
国籍	gj	Varchar2(10)	
客户申请日期	Sq_date	date	
客户状态	status	Char(1)	0：正常，1：禁用
客户级别	Cust_level	Char(1)	0：普通级别，1：VIP 级别
信用状态	Credet_statu	Number(1)	0：良好，1：违约，2：优秀
账单日期	Zd_date	date	客户固定账单日期每月几号
账单邮寄地址	Zdyj_addr	Varchar2(30)	
账单邮寄方式	Zdyj_fs	Char(1)	0：纸质账单，1：电子账单

（2）账户信息表

账户信息表存储客户基本的账户信息，是对信用卡账户的描述和信息记录，如表 10−2 所示。

表 10−2　账户信息表（account_info）

字　段　名	英文字段名	类　　型	备　　注
客户号	Custer_id	char(8)	客户标识号
账号	account_id	Char(16)	
卡类	Card_type	Char(10)	
姓名	Name	Varchar2(10)	客户姓名
账单日	zd_date	date	
分行号	Branch_id	Char(9)	
开卡日期	Open_date	date	信用卡开卡日期
账户状态	Zh_status	Number(1)	0：正常，1：销户
卡状态	status	Char(1)	0：正常，1：禁用
卡级别	Card_level	Char(1)	0：普通级别，1：VIP 级别
账户活跃度	hy_flag	Char(1)	0：一般，1：很活跃，3：不经常用此卡
每天取现额度	Ev_ed	Number(9)	
宽限期	Kx_days	number(4)	

（3）客户卡信息

客户卡信息记录客户用卡情况，如表 10−3 所示。

表 10−3　卡信息表（card_info）

字　段　名	英文字段名	类　　型	备　　注
账号	CARD_ID	Char(16)	
客户号	Custer_id	char(8)	客户标识号
卡号	Carder_id	Char(16)	
卡片类型	Card_type	Char(10)	
姓名	Name	Varchar2(10)	客户姓名
主附卡标识	Master_type	Char(1)	0：主卡，1：附卡
分行号	Branch_id	Char(9)	
客户额度	ed	Number(9)	卡信用额度
临时额度	Ls_ed	Number(9)	根据需要临时调整额度
调整日期	tz_date	date	临时额度调整日期
取现额度	qx_ed	Number(9)	信用卡提取现金最高额度
分期额度	Fq_ed	Number(9)	分期还款额度
上期还款额度	sqhk_ed	Number(9)	
上期还款日期	sqhk_date	date	
本期还款额度	Bqhk_ed	Number(9)	
本期还款日期	Bqhk_date	date	
欠款额度	Qk_ed	Number(9)	
欠款标识	Qk_type	Char(1)	0：不欠款，1：欠款
违约天数	Wy_days	Number(3)	
违约次数	Wy_times	Number(2)	
违约标志	Wy_flag	Char(1)	0：无违约，1：轻度违约，3：严重违约

（4）交易信息表

交易信息表记录客户刷信用卡交易的信息，如表 10-4 所示。

表 10-4　交易信息表（Transaction_inf）

字　段　名	英文字段名	类　　型	备　　注
账号	CARD_ID	Char(16)	
客户号	Custer_id	char(8)	客户标识号
卡号	Car_id	Char(16)	
卡片类型	Card_type	Char(10)	
姓名	Name	Varchar2(10)	客户姓名
消费额度	Xf_ed	Number(9)	
消费商户	Xf_sh	Varchar2(30)	
消费地点	Xf_addr	Varchar2(30)	
消费时间	xf_date	date	
消费类型	xf_type	char(1)	1：购物，2：取现金
本期消费累计笔数	bqXf_ljbs	Number(4)	
消费手续费	Xf_sxf	Number(9)	
消费利息	xf_lx	Number(9,3)	
消费当前利率	xf_dqlv	Number(5,3)	
消费积分	xf_jf	Number(9)	
还款交易额度	hk_ed	date	
卡片状态	card_status	Char(1)	0：正常，1：封锁

（5）客户信用卡申请信息表

客户信用卡申请信息是客户申请某类信用卡时所填写的信息，通过此信息可以查询客户的信用信息有无违约记录，如表 10-5 所示。

表 10-5　信用卡申请表（Apply_credit）

字　段　名	英文字段名	类　　型	备　　注
申请编号	Aplication_bh	Varchar2(8)	申请标识号
申请日期	Application_dte	date	
姓名	Apl_name	Varchar2(10)	
出生日期	Birth_dte	date	
性别	sex	char(1)	F：女，M：男
证件类型	zj_type	char(1)	
证件号码	zj_id	Varchar2(18)	
学历	education	Varchar2(10)	
婚姻状况	Marr_status	Char(1)	N：未婚，Y：已婚
手机号码	mobile	char(11)	
现住地址	Xz_addr	Varchar2(30)	
电话	phone	Varchar2(12)	

（续）

字　段　名	英文字段名	类　　型	备　　注
工作单位	work_dw	Varchar2(30)	
现单位工作年限	Xz_work_y	Interger(2)	
职称	zc	Varchar2(16)	
年收入	Year_earn	Number(9)	
工作年限	gz_year	Number(2)	
申请额度	Sq_ed	Number(9)	
父亲姓名	father	Varchar2(10)	
母亲姓名	mother	Varchar2(10)	
父母联系方式	Fm_phone	Varchar2(12)	
个人征信	zx	Char(1)	0：无违约记录，1：有违约记录
审批人编号	Sp_bh	Char(8)	
审批人姓名	Sp_name	Varchar2(8)	
评分值	Pf_value	Integer(3)	
评分结果	Pf_jg	Char(1)	0：一般，1：有风险，2：良好，3：优秀
建议额度	Jy_ed	Number(9)	

（6）客户积分信息表

客户积分是对客户刷卡送积分，是刺激刷卡消费的一种营销方式。通过对客户行为、交易数据，根据市场营销计划，设定相应规则，进行分类汇总形成积分并进行回馈客户，客户积分表，如表 10-6 所示。

表 10-6　客户积分表（cust_Integral）

字　段　名	英文字段名	类　　型	备　　注
账号	Card_id	Char(16)	
客户号	Custer_id	char(8)	客户标识号
卡号	Ca_id	Char(16)	
姓名	Name	Varchar2(10)	客户姓名
币种	bz_type	char(1)	B：人民币，W：外币
消费额度	Xf_ed	number(9)	
积分类型	Jf_type	char(1)	1：消费积分，2：取现积分
总积分	jf	Number(9)	
积分计算规则	Jfjs_rule	Number(1)	1：普通消费 1 元为 2 分，2：取现 1 元为 1 分
本期兑换积分	bqdh_jf	Number(9)	
本期积分	bq_jf	Number(9)	
兑换类型	dh_lx	Char(1)	W：换物品，m：换钱
本期奖励积分	bqjl_jf	Number(9)	
本期奖励原因	bqjl_yx	Varchar2(30)	

（7）客户消费信息表

客户消费信息表存储客户消费信息的记录，记录客户每笔消费交易的详细信息，如表 10-7 所示。

表 10-7　客户消费行为表（Consumer_behavior）

字　段　名	英文字段名	类　　型	备　　注
客户号	Custer_id	char(8)	客户标识号
卡号	C_id	Char(16)	
卡片类型	Card_type	Char(10)	
姓名	Name	Varchar2(10)	客户姓名
消费额度	Xf_ed	Number(9)	
消费商户	Xf_sh	Varchar(9)	
消费地点	Xf_addr	Varchar2(30)	
消费时间	xf_date	date	
消费类型	xf_type	char(1)	1：购物，2：取现金
本期消费累计笔数	Xf_bs	Number(4)	
每天消费笔数	Days_bs	Number(3)	
每天消费额度	Days_ed	Number(9)	
每月消费笔数	Month_bs	Number(3)	
每月消费额度	Month_ed	Number(9)	
每年消费笔数	Year_bs	Number(3)	
每年消费额度	Year_ed	Number(9)	

（8）客户风险信息表

客户风险信息表存储信用卡客户有风险和违约客户的信息，是记录客户不良信息的表，便于减少银行损失，提高风险预警。风险管理是识别、防范和控制银行卡申办和使用过程中的各种风险，通过对客户的资信评估，确定信用等级、分析透支情况、降低透支风险等，如表 10-8。

表 10-8　风险信息表（risk_info）

字　段　名	英文字段名	类　　型	备　　注
账号	Card_id	Char(16)	
客户号	Custer_id	char(8)	客户标识号
卡号	Carder_id	Char(16)	
卡片类型	Card_type	char(1)	
姓名	Name	Varchar2(10)	客户姓名
违约类型	Wy_type	char(1)	A：违约 15 天以内没有还款，B：违约 15～30 天，C：违约 30 天以上
欠款额度	qk_ed	number(9)	
欠款原因	qk_yx	Varchar2(30)	
账单日	zd_date	date	
最后还款日	Zhhk_date	date	
实际还款日	Sjhk_date	date	
违约天数	wy_days	Number(6)	

（9）客户催收信息表

催收系统是对催收的账户和客户资料导入到这个所谓的“催收管理系统”中，然后由催收主管分配任务，根据违约性质，对催收行为分级，一般催收级别为短信催、电话催、人工上门催、信函催等催收方式，能记录催收员的催收行为和催收反馈记录。通过客户催收信息表记录对违约客户的催收的情况，如表 10-9 所示。

表 10-9　催收信息表（collection_info）

字　段　名	英文字段名	类　　型	备　　注
催收案件编号	cs_id	char(10)	
账号	acount_id	Char(16)	
客户姓名	name	Varchar2(10)	
违约类型	Wy_type	char(1)	A：违约 15 天以内没有还款，B：违约 15～30 天，C：违约 30 天以上
欠款额度	qk_ed	number(9)	
欠款原因	qk_yx	Varchar2(30)	
催收级别	cs_level	Char(1)	D：短信催收，C：电话催收，B：信函催收，A：上门催收
催收到账额度	csdz_ed	Number(9,2)	
催收日期	cs_date	date	
是否停止催收	Sftz_cs	Char(1)	N：继续催收，Y：停止催收
停止催收日期	Tzcs_date	date	
催收员编号	Csy_id	Char(8)	
催收员姓名	Cs_name	Varchar2(10)	
催收天数	Cs_days	NUMBER(8)	

（10）客户营销信息表

客户营销信息表是对客户刷卡消费进行回馈的一种方式，对每月消费额度超过一定额度或每月消费笔数大于某个值给予优惠或送积分，如表 10-10 所示。

表 10-10　营销信息表（Marketing_info）

字　段　名	英文字段名	类　　型	备　　注
营销编号	yx_id	char(8)	
营销类型	yx_type	Char(1)	A：节日营销刷卡送积分；B：对客户生日的当天营销购买物品优惠；C：刷卡每月额度超过 2 000 元，送积分 500 分；D：营销发起期间刷卡优惠活动
营销分行号	yx_branchid	Char(8)	
营销地点	Yx_addr	Varchar2(30)	
营销员姓名	Yx_Name	Varchar2(10)	发起此活动的负责人姓名
营销员编号	yxy_id	char(8)	
营销合作商户	yxhz_sh	Varchar(30)	
营销产品	yx_cp	Varchar2(30)	
营销内部评价	yx_pj	Varchar2(30)	

理解以客户为中心的信用卡管理系统表与表的关系，可以帮助理解数据仓库内部表之间关联设计星形模式或雪花形模式的设计。表与表之间如何关联，也是前端查询时需要考虑的。合适的关联方式可以查询出用户需要的查询信息，同时也提高了前端应用的效率。

3．ETL数据装载方式

数据仓库的核心环节是 ETL 过程，这是数据仓库的地基，因此 ETL 提取外围数据到目标数据仓库的转化规则需要合理的设计，否则影响 ETL 处理数据的效率。其处理策略如下：

（1）全量数据装载

数据仓库数据表中只包括最新的数据，每次装载前先删除原有表数据，然后完全装载最新的源数据。这种装载模式称为全量数据装载，数据抽取程序抽取源数据中的所有记录，在装载前，将数据仓库中的目标数据表清空，然后装载所有记录。为提高清空目标表的数据速度，一般采用 truncate 清空目标数据表。对于全量数据一般数据量比较大，其设计 ETL 处理过程时一般先把数据装载到数据仓库的临时目标表中，然后再通过数据库的存储过程 merge 到目标表中，这样减少了库系统的问题，提高了 ETL 装载效率。对于全量数据，可以先清空表数据，然后再装载数据到目标表中。

（2）增量数据装载

对于增量数据装载，根据实际需求，每一条记录是一个新的事件，相互之间没有必然的联系，新记录不是对原有记录数值的变更，记录包括时间字段，可以通过时间字段将新增数据抽取出来装载到数据仓库目标表中。一般先对目标表根据当天日期进行一次删除操作，然后再追加数据到目标表中。

（3）镜像增量

数据仓库中的数据具有生效日期字段以保存数据的历史数据信息，而源数据不保留历史数据并且每天都可能更新。因此，只能将新的镜像数据与上次装载的数据的镜像进行比较，找出变更部分，更新历史数据被更新记录的生效终止日期，并添加变更后的数据。大多数据源数据中需要保存历史信息的维表。

（4）含索引表数据装载

实际 ETL 过程中会遇到有索引表的数据装载，一般选择 LOAD 方式。

4．数据抽取、清洗、转换、装载到目标数据库

数据仓库 ETL 处理过程是整个数据仓库中最重要的环节。对于外围数据的装载一般先把数据装载到目标数据仓库的临时表，而不是直接装载到目标表中，因为目标表很多都带有索引，如果直接装载到目标表中，索引会大大影响数据装载的效率。

本案例 ETL 过程的实现以 SAS 工具和 DataStage 工具分别装载实现同样的功能，目的是让读者学习 SAS 工具。

本案例 SAS 工具的应用可以选择 UNIX 环境，也可以选择 Windows 环境，为便于学习，SAS 工具选择 Windows 下的环境，数据仓库应用 ORACLE10G，登录数据库的用户名：chiran，密码：chiran，数据仓库实例：ORCL。

（1）客户信息表 cust_info 装载操作步骤

1）首先登录数据仓库 ORCL，创建 cust_info 表，其创建表语句如下：

```
-- Create table
```

```
create   table   CUST_INFO
(
  CUSTER_ID          CHAR(8),
  CARD_ID            CHAR(16),
  NAME               VARCHAR2(10),
  SEX                CHAR(1),
  ID                 VARCHAR2(18),
  BIRTH              DATE,
  MARITAL_STATUS CHAR(1),
  XL                 VARCHAR2(10),
  ZW                 VARCHAR2(12),
  WORK_YEAR          NUMBER(2),
  ADDR_1             VARCHAR2(30),
  ADDR_2             VARCHAR2(30),
  ZONE               VARCHAR2(10),
  MOBILE             CHAR(11),
  PHONE              VARCHAR2(12),
  E_MAIL             VARCHAR2(30),
  GJ                 VARCHAR2(10),
  SQ_DATE            DATE,
  STATUS             CHAR(1),
  CUST_LEVEL         CHAR(1),
  CREDET_STATU       NUMBER(1),
  ZD_DATE            DATE,
  ZDYJ_ADDR          VARCHAR2(30),
  ZDYJ_FS            CHAR(1)
)
tablespace TBS_CUST1;
```

【注意】 创建表语句中要把 tablespace 对应表空间 TBS_CUST1 换成读者自己的数据仓库对应的表空间。

2）外部数据文件 Cust.dat 装载到目标表 cust_info 的 SAS 程序如下：

```
  LIBNAME   qhlj   oracle   user=chiran   password=chiran   path=orcl;
/*建立连接数据库的逻辑库*/
LIBNAME   qh   'd:\qh'; /*定义逻辑库*/
%let   lj= D:\qh\cust;/*定义外部文件路径*/
%let   gsm=.dat; /*文件格式定义*/
%let   filename= "&lj&gsm";
DATA   qh.custinf   (drop=birth_1 sq_date_1 zd_date_1 );   /*数据集存储到指定逻辑库*/
     Infile   &filename   dlm='|'   dsd   missover   lrecl=269 ;
/*此处用 dlm='|'分隔符参数读取分隔文件，文件记录长度超过 256 用 lrecl=指定长度*/
          input   custer_id        :$8.
                  card_id          :$16.
                  name             :$10.
                  sex              :$1.
                  id               :$18.
```

```
                birth_1              :$8.
                marital_status     :$1.
                xl                    :$10.
                zw                    :$12.
                work_year          :2.
                addr_1              :$30.
                addr_2              :$30.
                zone                 :$10.
                mobile              :$12.
                phone                :$12.
                e_mail              :$30.
                gj                    :$10.
                sq_date_1         :$8.
                status               :$1.
                cust_level         :$1.
                credet_statu      :1.
                zd_date_1          :$8.
                zdyj_addr         :$30.
                zdyj_fs             :$1.
                ;
length     birth    8;
     if     birth_1< '19771231'    then    birth=input('19771231',anydtdte8.);
      else      do;
       if     birth_1> '21001231'    then    birth=input('21001231',anydtdte8.);
           else      birth =input(birth_1,anydtdte8.);
     end;
     length     sq_date    8;
     if     sq_date_1< '19771231'    then    sq_date=input('19771231',anydtdte8.);
      else    do;
       if     sq_date_1> '21001231'    then    sq_date=input('21001231',anydtdte8.);
           else    sq_date =input(sq_date_1,anydtdte8.);
     end;
     length    zd_date    8;
     if     zd_date_1< '19771231'    then    zd_date=input('19771231',anydtdte8.);
      else    do;
       if    zd_date_1> '21001231'    then    zd_date=input('21001231',anydtdte8.);
           else    zd_date =input(zd_date_1,anydtdte8.);
     end;
run;
proc    append    base=qhlj.cust_info
     ( bulkload=no
       dbsastype=(
       birth                 = 'date'
       work_year                 ='numeric'
       sq_date              = 'date'
       zd_date              = 'date'
```

```
        credet_statu      ='numeric'
    )
    nullchar=no
    nullcharval=" "
    )
    data=qh.custinf;
    run;
```

3）登录数据仓库，执行查询语句 select * from cust_info，查询信息表 cust_info，信息显示部分截图如图 10-2 所示。

	CUSTER_ID	CARD_ID	NAME	SEX	ID	BIRTH	MARITAL_STATUS	XL	ZW
1	10000001	1234567892345671	蒋为明	M	101000000000000001	1982-3-23	Y	研究生	教师
2	10000007	1234567892345677	王中	M	101000000000000007	1982-3-29	Y	本科	经理
3	10000008	1234567892345678	宋玉琴	M	101000000000000008	1982-3-11	Y	本科	教师

图 10-2　cust_info 表信息显示

（2）账户信息表 account_info 装载操作步骤

1）登录数据仓库 ORCL，创建 account_info 表，其创建表语句如下：

```
-- Create table
create   table   ACCOUNT_INFO
(
   CUSTER_ID    CHAR(8),
   ACCOUNT_ID CHAR(16),
   CARD_ID       CHAR(16),
   CARD_TYPE    CHAR(10),
   NAME            VARCHAR2(10),
   ZD_DATE        DATE,
   BRANCH_ID    CHAR(9),
   OPEN_DATE    DATE,
   ZH_STATUS    NUMBER,
   STATUS          CHAR(1),
   CARD_LEVEL CHAR(1),
   HY_FLAG        CHAR(1),
   EV_ED            NUMBER(9),
   KX_DAYS        NUMBER(4)
)
tablespace   TBS_CUST2;
```

2）外部数据文件 account.dat 装载到目标表 account_info 的 SAS 程序如下：

```
LIBNAME   qhlj   oracle   user=chiran   password=chiran   path=orcl;
/*建立连接数据库的逻辑库*/
LIBNAME   qh   'd:\qh'; /*定义逻辑库*/
%let   lj= D:\qh\account;/*定义外部文件路径*/
%let   gsm=.dat; /*文件格式定义*/
```

```
%let   filename= "&lj&gsm";
DATA   qh.account   (drop=zd_date_1 open_date_1);   /*数据集存储到指定逻辑库*/
       Infile   &filename   dlm='|'   dsd   missover   lrecl=102 ;
/*此处用 dlm='|'分隔符参数读取分隔文件，文件记录长度超过 256 用 lrecl=指定长度*/
 Input   custer_id      :$8.
         account_id     :$16.
         card_id        :$16.
         card_type      :$10.
         name           :$10.
         zd_date_1      :$8.
         branch_id      :$9.
         open_date_1    :$8.
         zh_status       :1.
         status         :$1.
         card_level      :$1.
         hy_flag         :$1.
         ev_ed          :9.
         kx_days         :4.
               ;
length   zd_date   8;
       if    zd_date_1< '19771231'   then   zd_date=input('19771231',anydtdte8.);
        else   do;
         if   zd_date_1> '21001231'   then   zd_date=input('21001231',anydtdte8.);
            else     zd_date =input(zd_date_1,anydtdte8.);
       end;
       length   open_date   8;
       if    open_date_1< '19771231'   then   open_date=input('19771231',anydtdte8.);
        else   do;
         if   open_date_1> '21001231'   then   open_date=input('21001231',anydtdte8.);
            else   open_date =input(open_date_1,anydtdte8.);
       end;
run;
proc   append   base=qhlj.account_info
       ( bulkload=no
         dbsastype=(
         zd_date              = 'date'
         open_date                  = 'date'
         zh_status            = 'numeric'
         ev_ed                = 'numeric'
         kx_days    = 'numeric'
       )
       nullchar=no
       nullcharval=" "
       )
       data=qh.account;
       run;
```

3）登录数据仓库，执行查询语句 select * from account_info，查询信息表 cust_info，信息显示部分截图如图 10-3 所示。

	CUSTER_ID	ACCOUNT_ID	CARD_ID	CARD_TYPE	NAME	ZD_DATE	BRANCH_ID
1	10000001	2234567892345676	1234567892345671	银联	蒋为明	2008-6-8	123456789
2	10000007	8234567892345673	1234567892345677	MASTER	王中	2008-4-8	234567818
3	10000008	9234567892345675	1234567892345678	VISA	宋玉琴	2008-4-8	312456786

图 10-3　account_info 表信息显示

（3）卡信息表 card_info 装载操作步骤

1）登录数据仓库 ORCL，创建 card_info 表，其创建表语句如下：

```
  -- Create table
create   table   CARD_INFO
(
  CARD_ID         CHAR(16),
  CUSTER_ID       CHAR(8),
  CARDER_ID       CHAR(16),
  CARD_TYPE       CHAR(10),
  NAME            VARCHAR2(10),
  MASTER_TYPE CHAR(1),
  BRANCH_ID       CHAR(9),
  ED              NUMBER(9),
  LS_ED           NUMBER(9),
  TZ_DATE         DATE,
  QX_ED           NUMBER(9),
  FQ_ED           NUMBER(9),
  SQHK_ED         NUMBER(9),
  SQHK_DATE       DATE,
  BQHK_ED         NUMBER(9),
  BQHK_DATE       DATE,
  QK_ED           NUMBER(9),
  QK_TYPE         CHAR(1),
  WY_DAYS         NUMBER(3),
  WY_TIMES        NUMBER(2),
  WY_FLAG         CHAR(1)
)
tablespace TBS_CUST3;
```

2）外部数据文件 card.dat 装载到目标表 card_info 的 SAS 程序如下：

```
LIBNAME   qhlj   oracle   user=chiran   password=chiran   path=orcl;
/*建立连接数据库的逻辑库*/
LIBNAME   qh 'd:\qh'; /*定义逻辑库*/
%let   lj= D:\qh\card;/*定义外部文件路径*/
%let   gsm=.dat; /*文件格式定义*/
```

```
%let   filename= "&lj&gsm";
DATA   qh.cardinfo   (drop=tz_date_1 sqhk_date_1 bqhk_date_1);   /*数据集存储到指定逻辑库*/
       Infile   &filename   dlm='|'   dsd   missover   lrecl=164 ;
/*此处用 dlm='|'分隔符参数读取分隔文件，文件记录长度超过 256 用 lrecl=指定长度*/
input   card_id          :$16.
          custer_id        :$8.
          carder_id        :$16.
          card_type        :$10.
          name             :$10.
          master_type      :$1.
          branch_id        :$9.
          ed               :9.
          ls_ed            :9.
          tz_date_1        :$8.
          qx_ed            :9.
          fq_ed            :9.
          sqhk_ed          :9.
          sqhk_date_1      :$8.
          bqhk_ed          :9.
          bqhk_date_1      :$8.
          qk_ed            :9.
          qk_type          :$1.
          wy_days          :3.
          wy_times         :2.
          wy_flag          :$1.
                           ;
length   tz_date   8;
       if    tz_date_1< '19771231'   then   tz_date=input('19771231',anydtdte8.);
        else   do;
         if   tz_date_1> '21001231'   then   tz_date=input('21001231',anydtdte8.);
             else     tz_date =input(tz_date_1,anydtdte8.);
       end;
       length   sqhk_date   8;
       if    sqhk_date_1< '19771231'   then   sqhk_date=input('19771231',anydtdte8.);
        else   do;
         if   sqhk_date_1> '21001231'   then   sqhk_date=input('21001231',anydtdte8.);
             else   sqhk_date=input(sqhk_date_1,anydtdte8.);
       end;
       length   bqhk_date   8;
       if    bqhk_date_1< '19771231'   then   bqhk_date=input('19771231',anydtdte8.);
        else   do;
         if   bqhk_date_1> '21001231'   then   bqhk_date=input('21001231',anydtdte8.);
             else     bqhk_date=input(bqhk_date_1,anydtdte8.);
       end;
run;
proc   append   base=qhlj.card_info
```

```
( bulkload=no
  dbsastype=(
  ed            ='numeric'
  ls_ed             ='numeric'
  tz_date       = 'date'
  qx_ed             ='numeric'
  fq_ed             ='numeric'
  sqhk_ed      ='numeric'
  sqhk_date    = 'date'
  bqhk_ed      ='numeric'
  bqhk_date            = 'date'
  qk_ed           = 'numeric'
  wy_days   ='numeric'
  wy_times         ='numeric'
)
nullchar=no
nullcharval=" "
)
data=qh.cardinfo;
run;
```

3）登录数据仓库，执行查询语句 select * from card_info，查询信息表 card_info，信息显示部分截图如图 10-4 所示。

	CARD_ID	CUSTER_ID	CARDER_ID	CARD_TYPE	NAME	MASTER_TYPE	BRANCH_ID	ED
1	2234567892345676	10000001	1234567892345671	银联	蒋为明	0	123456789	7000
2	8234567892345673	10000007	1234567892345677	MASTER	王中	1	234567818	5000
3	9234567892345675	10000008	1234567892345678	VISA	宋玉琴	0	312456786	8000

图 10-4　card_info 表信息显示

（4）交易信息表 Transaction_inf 装载操作步骤

1）登录数据仓库 ORCL，创建 transaction_inf 表，其创建表语句如下：

```
-- Create table
create   table   TRANSACTION_INF
(
  CARD_ID        CHAR(16),
  CUSTER_ID      CHAR(8),
  CAR_ID         CHAR(16),
  CARD_TYPE      CHAR(10),
  NAME           VARCHAR2(10),
  XF_ED          NUMBER(9),
  XF_SH          VARCHAR2(30),
  XF_ADDR        VARCHAR2(30),
  XF_DATE        DATE,
  XF_TYPE        CHAR(1),
  BQXF_LJBS      NUMBER(4),
```

```
  XF_SXF          NUMBER(9),
  XF_LX           NUMBER(9,3),
  XF_DQLV         NUMBER(5,3),
  XF_JF           NUMBER(9),
  HK_ED           NUMBER(9),
  CARD_STATUS CHAR(1)
)
tablespace TBS_CUST1;
```

2）外部数据文件 transaction.dat 装载到目标表 transaction_inf 的 SAS 程序如下：

```
LIBNAME   qhlj   oracle   user=chiran   password=chiran   path=orcl;
/*建立连接数据库的逻辑库*/
LIBNAME   qh 'd:\qh'; /*定义逻辑库*/
%let   lj= D:\qh\transaction;/*定义外部文件路径*/
%let   gsm=.dat; /*文件格式定义*/
%let   filename= "&lj&gsm";
DATA   qh.transaction   (drop=xf_date_1 );   /*数据集存储到指定逻辑库*/
     Infile   &filename   dlm='|'   dsd   missover   lrecl=184 ;
input    Card_id            :$16.
         Custer_id          :$8.
         Car_id             :$16.
         Card_type          :$10.
         Name               :$10.
         Xf_ed              :9.
         Xf_sh              :$30.
         Xf_addr            :$30.
         xf_date_1          :$8.
         xf_type            :$1.
         bqXf_ljbs          :4.
         Xf_sxf             :9.
         xf_lx              :9.3
         xf_dqlv            :5.3
         xf_jf              :9.
         hk_ed              :9.
         card_status        :$1.

                  ;
length   xf_date   8;
     if    xf_date_1< '19771231'   then    xf_date=input('19771231',anydtdte8.);
      else   do;
       if   xf_date_1> '21001231'   then   xf_date=input('21001231',anydtdte8.);
           else     xf_date =input(xf_date_1,anydtdte8.);
     end;
run;
proc   append   base=qhlj.transaction_inf
     ( bulkload=no
```

```
    dbsastype=(
    Xf_ed                 ='numeric'
    xf_date        = 'date'
    bqXf_ljbs                ='numeric'
    xf_lx             ='numeric'
    xf_dqlv       ='numeric'
    xf_jf           ='numeric'
    hk_ed          ='numeric'
  )
  nullchar=no
  nullcharval=" "
  )
  data=qh.transaction;
  run;
```

3）登录数据仓库，执行查询语句 select * from transaction_inf，查询信息表 transaction_inf，交易表显示部分交易信息截图如图 10-5 所示。

NAME	XF_ED	XF_SH	XF_ADDR	XF_DATE	XF_TYPE	BQXF_LJBS	XF_SXF	XF_LX	XF_DQLV
蒋为明	4000	北京华联商场	北京	2011-3-23	1	30	0	15.300	0.350
王中	3000	青岛大瑞发	青岛	2011-2-15	1	20	10	0.000	0.250
宋玉琴	1500	法国家乐福	法国	2011-3-10	2	15	30	23.600	0.260

图 10-5　交易信息表数据显示

（5）客户消费行为信息表 Consumer_behavior 装载操作步骤

1）登录数据仓库 ORCL，创建 Consumer_behavior 表，其创建表语句如下：

```
-- Create table
create   table   CONSUMER_BEHAVIOR
(
  CARD_ID      CHAR(16),
  CUSTER_ID CHAR(8),
  C_ID          CHAR(16),
  CARD_TYPE CHAR(10),
  NAME          VARCHAR2(10),
  XF_ED         NUMBER(9),
  XF_SH         VARCHAR2(30),
  XF_ADDR      VARCHAR2(30),
  XF_DATE      DATE,
  XF_TYPE      CHAR(1),
  XF_BS         NUMBER(4),
  DAYS_BS      NUMBER(3),
  DAYS_ED      NUMBER(9),
  MONTH_BS    NUMBER(3),
  MONTH_ED    NUMBER(9),
  YEAR_BS      NUMBER(3),
```

```
   YEAR_ED     NUMBER(9)
)
tablespace TBS_CUST2;
```

2）外部数据文件 consumer.dat 装载到目标表 Consumer_behavior 的 SAS 程序如下：

```
LIBNAME   qhlj   oracle   user=chiran   password=chiran   path=orcl;
/*建立连接数据库的逻辑库*/
LIBNAME   qh 'd:\qh'; /*定义逻辑库*/
%let   lj= D:\qh\consumer;/*定义外部文件路径*/
%let   gsm=.dat; /*文件格式定义*/
%let   filename= "&lj&gsm";
DATA   qh.consumer   (drop=xf_date_1);   /*数据集存储到指定逻辑库*/
      Infile   &filename   dlm='|'   dsd   missover   lrecl=178 ;
/*此处用 dlm='|'分隔符参数读取分隔文件，文件记录长度超过 256 用 lrecl=指定长度*/
input    card_id         :$16.
         custer_id       :$8.
         c_id            :$16.
         card_type       :$10.
         name            :$10.
         xf_ed           :9.
         xf_sh           :$30.
         xf_addr         :$30.
         xf_date_1       :$8.
         xf_type         :$1.
         xf_bs           :4.
         days_bs         :3.
         days_ed         :9.
         month_bs        :3.
         month_ed        :9.
         year_bs         :3.
         year_ed         :9.
                    ;
length   xf_date   8;
      if    xf_date_1< '19771231'   then   xf_date=input('19771231',anydtdte8.);
       else   do;
        if   xf_date_1> '21001231'   then   xf_date=input('21001231',anydtdte8.);
            else     xf_date =input(xf_date_1,anydtdte8.);
      end;
run;
proc   append   base=qhlj.consumer_behavior
      ( bulkload=no
        dbsastype=(
        xf_ed             ='numeric'
        xf_date       = 'date'
        xf_bs           ='numeric'
        days_bs         ='numeric'
```

```
        days_ed       ='numeric'
        month_bs    = 'numeric'
        month_ed            ='numeric'
        year_bs            = 'numeric'
        year_ed     ='numeric'
    )
    nullchar=no
    nullcharval=" "
    )
    data=qh.consumer;
    run;
```

3）登录数据仓库，执行查询语句 select * from consumer_behavior，查询信息表 consumer_behavior，消费行为表显示部分消费信息截图如图 10-6 所示。

XF_SH	XF_ADDR	XF_DATE	XF_TYPE	XF_BS	DAYS_BS	DAYS_ED	MONTH_BS
北京华联商场	北京	2011-3-23	1	30	10	600	26
青岛大瑞发	青岛	2011-2-15	1	20	5	500	30
法国家乐福	法国	2011-3-10	2	10	6	300	28

图 10-6　消费行为表数据显示

（6）催收信息表 collection_info 装载操作步骤

1）登录数据仓库 ORCL，创建 collection_info 表，其创建表语句如下：

```
-- Create table
create   table   COLLECTION_INFO
(
  ACCOUNT_ID CHAR(16),
  CS_ID          CHAR(10),
  ACOUNT_ID    CHAR(16),
  NAME            VARCHAR2(10),
  WY_TYPE        CHAR(1),
  QK_ED           NUMBER(9),
  QK_YX           VARCHAR2(30),
  CS_LEVEL      CHAR(1),
  CSDZ_ED        NUMBER(9),
  CS_DATE        DATE,
  SFTZ_CS        CHAR(1),
  TZCS_DATE    DATE,
  CSY_ID          CHAR(8),
  CS_NAME        VARCHAR2(10),
  CS_DAYS        NUMBER(8)
)
tablespace   TBS_CUST3;
```

2）外部数据文件 collection.dat 装载到目标表 collection_info 的 SAS 程序如下：

```
LIBNAME   qhlj   oracle   user=chiran   password=chiran   path=orcl;
/*建立连接数据库的逻辑库*/
LIBNAME   qh   'd:\qh'; /*定义逻辑库*/
%let   lj= D:\qh\collection;/*定义外部文件路径*/
%let   gsm=.dat; /*文件格式定义*/
%let   filename= "&lj&gsm";
DATA   qh.collection   (drop=cs_date_1 tzcs_date_1);   /*数据集存储到指定逻辑库*/
       Infile   &filename   dlm='|'   dsd   missover   lrecl=145 ;
input    account_id      :$16.
           cs_id            :$10.
           acount_id        :$16.
           name             :$10.
           wy_type          :$1.
           qk_ed            :9.
           qk_yx            :$30.
           cs_level         :$1.
           csdz_ed          :9.
           cs_date_1        :$8.
           sftz_cs          :$1.
           tzcs_date_1      :$8.
           csy_id           :$8.
           cs_name          :$10.
           cs_days          :8.

                     ;
length   cs_date   8;
       if    cs_date_1< '19771231'   then   cs_date=input('19771231',anydtdte8.);
        else   do;
         if   cs_date_1> '21001231'   then   cs_date=input('21001231',anydtdte8.);
             else   cs_date =input(cs_date_1,anydtdte8.);
       end;

       length   tzcs_date 8;
       if    tzcs_date_1< '19771231'   then   tzcs_date=input('19771231',anydtdte8.);
        else   do;
         if    tzcs_date_1> '21001231'   then   tzcs_date=input('21001231',anydtdte8.);
             else    tzcs_date =input(tzcs_date_1,anydtdte8.);
       end;
run;
proc   append   base=qhlj.collection_info
       ( bulkload=no
         dbsastype=(
         qk_ed             ='numeric'
         csdz_ed          ='numeric'
         cs_date           = 'date'
         tzcs_date       ='date'
```

```
    cs_days       ='numeric'
  )
  nullchar=no
  nullcharval=" "
  )
  data=qh.collection;
  run;
```

3）登录数据仓库，执行查询语句 select * from collection_info，查询信息表 collection_info，催收表显示部分催收信息截图如图 10-7 所示。

	ACCOUNT_ID	CS_ID	ACCOUNT_ID	NAME	WY_TYPE	QK_ED	QK_YX	CS_LEVEL
1	1234567892345673	1010000011	6234567892345673	高为花	A	1000	忘记还款 ...	D
2	5234567892345676	1010000012	8234567892345676	刘小红	B	2000	没有钱 ...	C
3	8234567892345671	1010000013	9234567892345672	马卡凯	C	10800	恶意欠款 ...	A

图 10-7　催收表数据显示

（7）客户积分信息表 cust_Integral 装载操作步骤

1）登录数据仓库 ORCL，创建 cust_Integral 表，其创建表语句如下：

```
-- Create table
Create  table  CUST_INTEGRAL
(
  CARD_ID     CHAR(16),
  CUSTER_ID CHAR(8),
  CA_ID       CHAR(16),
  NAME         VARCHAR2(10),
  BZ_TYPE     CHAR(1),
  XF_ED       NUMBER(9),
  JF_TYPE     CHAR(1),
  JF           NUMBER(9),
  JFJS_RULE NUMBER(1),
  BQDH_JF     NUMBER(9),
  BQ_JF       NUMBER(9),
  DH_LX       CHAR(1),
  BQJL_JF     NUMBER(9),
  BQJL_YX     VARCHAR2(30)
)
tablespace TBS_CUST1;
```

2）外部数据文件 integral.dat 装载到目标表 cust_Integral 的 SAS 程序如下：

```
LIBNAME  qhlj  oracle  user=chiran  password=chiran  path=orcl;
/*建立连接数据库的逻辑库*/
LIBNAME  qh 'd:\qh'; /*定义逻辑库*/
%let  lj= D:\qh\integral;/*定义外部文件路径*/
%let  gsm=.dat; /*文件格式定义*/
```

```
%let   filename= "&lj&gsm";
DATA   qh.integral ;   /*数据集存储到指定逻辑库*/
       Infile   &filename   dlm='|'   dsd   missover   lrecl=129 ;
input    card_id          :$16.
          custer_id       :$8.
          ca_id            :$16.
          name             :$10.
          bz_type          :$1.
          xf_ed            :9.
          jf_type          :$1.
          jf                :9.
          jfjs_rule        :1.
          bqdh_jf          :9.
          bq_jf            :9.
          dh_lx            :$1.
          bqjl_jf           :9.
          bqjl_yx          :$30.
                     ;
run;
proc   append   base=qhlj.cust_integral
       ( bulkload=no
         dbsastype=(
         xf_ed               ='numeric'
         jf         ='numeric'
          jfjs_rule                ='numeric'
         bqdh_jf         ='numeric'
         bq_jf               ='numeric'
         bqjl_jf         ='numeric'
       )
       nullchar=no
       nullcharval=" "
       )
       data=qh.integral;
       run;
```

3）登录数据仓库，执行查询语句 select * from cust_Integral，查询信息表 cust_Integral，客户积分表显示部分积分信息截图如图 10-8 所示。

NAME	BZ_TYPE	XF_ED	JF_TYPE	JF	JFJS_RULE	BQDH_JF	BQ_JF	DH_LX	BQJL_JF	BQJL_YX
杨新雨	B	8000	1	19000	1	300	16000	w	0	无奖励
赵小东	W	9000	2	10000	2	100	9000	m	300	客户生日
王东王	B	90800	1	21000	1	50	181600	w	100	节日刷卡消费

图 10-8　客户积分表数据显示

（8）信用卡申请信息表 Apply_credit 装载操作步骤

1）登录数据仓库 ORCL，创建 Apply_credit 表，其创建表语句如下：

```
Create    table    apply_credit
(
   aplication_bh      char(8),
   application_dte date,
   apl_name              varchar2(10),
   birth_dte             date,
   sex                       char(1),
   zj_type                char(1),
   zj_id                    varchar2(18),
   education            varchar2(10),
   marr_status          char(1),
   mobile                  char(11),
   xz_addr                varchar2(30),
   phone                   varchar2(12),
   work_dw                varchar2(30),
   xz_work_y             integer,
   zc                         varchar2(16),
   year_earn             number(9),
   gz_year                number(2),
   sq_ed                    number(9),
   father                  varchar2(10),
   mother                   varchar2(10),
   fm_phone               varchar2(12),
   zx                         char(1),
   sp_bh                    char(8),
   sp_name                 varchar2(8),
   pf_value              integer,
   pf_jg                    char(1),
   jy_ed                     number(9)
)
tablespace    tbs_cust2;
```

2）外部数据文件 apply.dat 装载到目标表 apply_credit 的 SAS 程序如下：

```
LIBNAME    qhlj    oracle    user=chiran    password=chiran    path=orcl;
/*建立连接数据库的逻辑库*/
LIBNAME    qh 'd:\qh'; /*定义逻辑库*/
%let    lj= D:\qh\apply;/*定义外部文件路径*/
%let    gsm=.dat; /*文件格式定义*/
%let    filename= "&lj&gsm";
DATA    qh.apply (drop=application_dte_1 birth_dte_1 );    /*数据集存储到指定逻辑库*/
        Infile    &filename    dlm='|'    dsd    missover    lrecl=248 ;
input    aplication_bh            :$8.
            application_dte_1    :$8.
            apl_name               :$10.
            birth_dte_1          :$8.
            sex                        :$1.
```

```
          zj_type              :$1.
          zj_id                :$18.
          education            :$10.
          marr_status          :$1.
          mobile               :$12.
          xz_addr              :$30.
          phone                :$12.
          work_dw              :$30.
          xz_work_y            :2.
          zc                   :$16.
          year_earn            :9.
          gz_year              :2.
          sq_ed                :9.
          father               :$10.
          mother               :$10.
          fm_phone             :$12.
          zx                   :$1.
          sp_bh                :$8.
          sp_name              :$8.
          pf_value             :3.
          pf_jg                :$1.
          jy_ed                :9.

                 ;
        length   application_dte 8;
        if    application_dte_1< '19771231' then application_dte=input('19771231',anydtdte8.);
         else   do;
          if   application_dte_1> '21001231' then application_dte=input('21001231',anydtdte8.);
              else   application_dte =input(application_dte_1,anydtdte8.);
        end;
        length   birth_dte 8;
        if    birth_dte_1<'19771231' then birth_dte=input('19771231',anydtdte8.);
         else   do;
          if   birth_dte_1> '21001231' then birth_dte=input('21001231',anydtdte8.);
              else   birth_dte =input(birth_dte_1,anydtdte8.);
        end;


run;
proc   append   base=qhlj.apply_credit
        ( bulkload=no
          dbsastype=(
          application_dte = 'date'
          birth_dte       = 'date'
          xz_work_y              ='numeric'
          year_earn              ='numeric'
```

```
    gz_year          ='numeric'
    sq_ed                   ='numeric'
    pf_value         ='numeric'
    jy_ed                   ='numeric'
)
nullchar=no
nullcharval=" "
)
data=qh.apply;
run;
```

3）登录数据仓库，执行查询语句 select * from apply_credit，查询信息表 apply_credit，客户申请表显示部分信息截图如图 10−9 所示。

	APLICATION_BH	APPLICATION_DTE	APL_NAME	BIRTH_DTE	SEX	ZJ_TYPE	ZJ_ID	EDUCATION	MARR_STATUS
1	12300101	2011-8-9	刘新峰	1981-6-23	m	0	88991236	本科	N
2	12300102	2011-8-10	阮小七	1983-11-21	m	1	101025198311215816	研究生	N
3	12300103	2011-8-7	盛春号	1978-3-16	f	1	380029197803163128	博士	Y

图 10−9　客户申请信息表数据显示

（9）风险信息表 risk_info 装载操作步骤

1）登录数据仓库 ORCL，创建 risk_info 表，其创建表语句如下：

```
-- Create table
create table RISK_INFO
(
  CARD_ID      CHAR(16),
  CUSTER_ID CHAR(8),
  CARDER_ID CHAR(16),
  CARD_TYPE CHAR(10),
  NAME          VARCHAR2(10),
  WY_TYPE      CHAR(1),
  QK_ED         NUMBER(9),
  QK_YX         VARCHAR2(30),
  ZD_DATE      DATE,
  ZHHK_DATE DATE,
  SJHK_DATE DATE,
  WY_DAYS      NUMBER(3)
)
tablespace TBS_CUST3;
```

2）外部数据文件 risk.dat 装载到目标表 risk_info 的 SAS 程序如下：

```
LIBNAME   qhlj   oracle   user=chiran   password=chiran   path=orcl;
/*建立连接数据库的逻辑库*/
LIBNAME   qh 'd:\qh'; /*定义逻辑库*/
%let   lj= D:\qh\risk;/*定义外部文件路径*/
%let   gsm=.dat; /*文件格式定义*/
```

```
%let    filename= "&lj&gsm";
DATA    qh.risk    (drop=zd_date_1 zhhk_date_1 sjhk_date_1);    /*数据集存储到指定逻辑库*/
        Infile    &filename    dlm='|'    dsd    missover ;
/*此处用 dlm='|'分隔符参数读取分隔文件，文件记录长度超过 256 用 lrecl=指定长度*/
             input    card_id          :$16.
                    custer_id              :$8.
                    carder_id              :$16.
                    card_type              :$10.
                    name                   :$10.
                    wy_type                :$1.
                    qk_ed                  :9.
                    qk_yx                  :$30.
                    zd_date_1              :$8.
                    zhhk_date_1            :$8.
                    sjhk_date_1            :$8.
                    wy_days                :3.


                              ;
length    zd_date    8;
        if     zd_date_1<'19771231' then zd_date=input('19771231',anydtdte8.);
         else    do;
           if zd_date_1> '21001231' then zd_date=input('21001231',anydtdte8.);
               else    zd_date =input(zd_date_1,anydtdte8.);
        end;
        length    zhhk_date    8;
        if    zhhk_date_1<'19771231' then zhhk_date=input('19771231',anydtdte8.);
         else    do;
           if    zhhk_date_1> '21001231' then zhhk_date=input('21001231',anydtdte8.);
               else    zhhk_date =input(zhhk_date_1,anydtdte8.);
        end;
        length    sjhk_date    8;
        if     sjhk_date_1<'19771231' then sjhk_date=input('19771231',anydtdte8.);
         else    do;
           if    sjhk_date_1> '21001231' then sjhk_date=input('21001231',anydtdte8.);
               else    sjhk_date =input(sjhk_date_1,anydtdte8.);
        end;
run;
proc    append    base=qhlj.risk_info
        ( bulkload=no
           dbsastype=(

           qk_ed                     ='numeric'
             zd_date           = 'date'
           zhhk_date                 = 'date'
           sjhk_date                 = 'date'
```

```
        wy_days  ='numeric'
    )
    nullchar=no
    nullcharval=" "
    )
    data=qh.risk;
    run;
```

3）登录数据仓库，执行查询语句 select * from risk_info，查询风险表 risk_info，风险信息表显示部分数据截图如图 10-10 所示。

CARDER_ID	CARD_TYPE	NAME	WY_TYPE	QK_ED	QK_YX	ZD_DATE	ZHHK_DATE	SJHK_DATE
1234567892341669	1	懂轻轻	A	3980	忘记还款	2011-3-8	2011-4-26	2011-4-29
1234567892341665	2	马东风	C	6500	恶意欠款	2011-6-12	2011-7-12	2011-8-23
1234567892341663	0	古天月	B	7800	没有钱	2011-5-9	2011-6-18	2011-9-16

图 10-10　风险信息表数据显示

（10）营销信息表 Marketing_info 装载操作步骤

1）登录数据仓库 ORCL，创建 Marketing_info 表，其创建表语句如下：

```
-- Create table
create table MARKETING_INFO
(
  YX_ID          CHAR(8),
  YX_TYPE        CHAR(1),
  YX_BRANCHID CHAR(8),
  YX_ADDR        VARCHAR2(30),
  YX_NAME        VARCHAR2(10),
  YXY_ID         CHAR(8),
  YXHZ_SH        VARCHAR2(30),
  YX_CP          VARCHAR2(30),
  YX_PJ          VARCHAR2(30)
)
tablespace TBS_CUST3;
```

2）外部数据文件 apply.dat 装载到目标表 Marketing_info 的 SAS 程序如下：

```
LIBNAME   qhlj   oracle   user=chiran   password=chiran   path=orcl;
/*建立连接数据库的逻辑库*/
LIBNAME   qh 'd:\qh'; /*定义逻辑库*/
%let   lj= D:\qh\market;/*定义外部文件路径*/
%let   gsm=.dat; /*文件格式定义*/
%let   filename= "&lj&gsm";
DATA   qh.market;   /*数据集存储到指定逻辑库*/
      Infile   &filename   dlm='|'   dsd   missover ;
/*此处用 dlm='|'分隔符参数读取分隔文件，文件记录长度超过 256 用 lrecl=指定长度*/
        input   yx_id          :$8.
```

```
                    yx_type         :$1.
                    yx_branchid :$8.
                    yx_addr         :$30.
                    yx_name         :$10.
                    yxy_id          :$8.
                    yxhz_sh         :$30.
                    yx_cp           :$30.
                    yx_pj           :$30.
                            ;
run;
proc   append   base=qhlj.marketing_info
        ( bulkload=no
        nullchar=no
        nullcharval=" "
        )
        data=qh.market;
        run;
```

3）登录数据仓库，执行查询语句 select * from Marketing_info，查询营销信息表 Marketing_info，营销信息表显示部分数据截图如图 10-11 所示。

	YX_ID	YX_TYPE	YX_BRANCHID	YX_ADDR	YX_NAME	YXY_ID	YXHZ_SH	YX_CP	YX_PJ
1	12010017	A	10101235	北京分行	张东小	10101001	大中电器	笔记本	刺激了消费，比较好
2	12010016	C	10101259	天津分行	刘春国	10221003	苏宁电器	数码相机	消费量大增，很好
3	12010013	D	10101268	山东分行	高庆美	10531237	华联	冬季服装	消费量一般

图 10-11　营销信息表数据显示

10.3　数据挖掘信贷风险案例

数据挖掘案例从数据挖掘业务定位、目标定位、数据准备、模型开发与算法应用以及 SAS/EM 工具的实现演示了整个数据挖掘的过程。

（1）数据挖掘业务定位

数据挖掘是为解决某个业务背景下的问题，伴随业务背景而提出的目标需求。因此理解业务需求是进行数据挖掘的第一步。现实世界中的业务分析根据行业和需求的不同，其业务定位也各有不同。

信贷风险案例分析：业务背景

随着我国经济的发展，我国信贷市场空间不断拓展。高速增长的信贷已成为当前银行信用风险管理的一个突出问题，在信贷过程中，银行和贷款者作为不同利益的市场主体，是一种委托人和代理人的关系，两者之间存在信息不对称。虽然信贷业务的发展提高了银行的经营收入，扩大了银行的规模，但信贷业务中存在的问题和风险也日益暴露出来。目前我国尚未建立起一套完备的信贷制度，银行缺乏征询和调查借款者资信的有效手段，银行难以对借款者的财产和还款意愿等资信状况做出正确判断，各种恶意欺诈行为时有发生，产生大量

不良贷款。

贷款者提出向银行贷款后，银行需要对贷款者的还款能力和信用状况等情况进行了解，贷款者可以是企业，也可以是个人。由于存在信息不对称，银行不能完全掌握贷款者的具体情况，贷款者可能伪造财务报表、提供虚假信息来骗取银行的信任，从而来获取银行贷款。在这种情况下，会产生逆向选择。在信贷市场上那些最有可能造成不利结果，即造成违约风险的贷款者，往往就是那些寻求资金最积极而且最有可能得到资金的贷款者，这样银行的贷款将会产生很大的风险，可能会产生不良贷款。贷款者分为两种，一种是有信誉的，另一种是没有信誉的，有信誉的贷款者会按时偿还银行贷款，而没有信誉的贷款者则不会偿还贷款。同时，提供贷款的银行也面临两种选择，即提供贷款和不提供贷款。

实际贷款过程中，存在着银行和贷款者的信息不对称，银行并不拥有贷款者的全部信息。在目前我国的信贷市场中，贷款者的信誉状况并不是很好，各种经济诈骗、合同不履行、经理人缺乏诚信、相互拖欠等现象层出不尽。贷款者可能采取一些行为，故意隐瞒对自己不利的信息来获取银行的贷款。银行为取得贷款者的真实信息，需要投入一定的成本来收集这些信息，当收集信息的成本过大时，银行的利润就会极大地降低，这时银行会要求贷款者提供抵押。抵押是指为担保债务的履行，债务人或者第三人不转移财产的占有，将该财产设定为担保物，债务人不履行到期债务或者发生当事人约定的实现担保物权的情形，债权人有权就该财产优先受偿。在贷款者提供抵押的情况下，银行的信贷风险会大大降低。

当前信贷市场逐渐放宽，可以说现在已经进入了一个信贷发展的时代，其贷款条件和限制比较宽松，这也就带来了收益与风险同时相伴相生。信贷违约客户逐渐增多，对银行业造成的损失是巨大的，银行业面临着巨大的信贷违约风险。为防范信贷违约风险，规范信贷流程，必须依靠高科技手段。

（2）数据挖掘目标定位

数据挖掘目标定位是根据业务需求而确定的分析目标，是业务需求的明确化定位。对业务需求的信息进行提取，明确数据挖掘的主题。

信贷风险案例分析：目标定位

根据业务需求分析，为防范信贷风险，根据银行信贷风险数据仓库中的数据，对申请贷款的个人和企业通过对历史信贷数据信息的分析，研究当前信贷风险状况，从而进行风险预警，同时也对当前信贷情况进行一个评估与分析，验证当前防范措施是否有效，为未来信贷风险的发展提出更加有效的防范措施，降低信贷违约给银行带来的损失，为银行业下一步风险提出更加可行性的防范措施。

建立信贷风险历史数据仓库，通过数据挖掘工具，为信贷风险提出如下措施：

1）科学化信贷管理制度，建立约束机制。增强信贷管理和决策科学化是管理贷款、防范贷款风险的基础性工作。必须严格执行贷款审贷分离制度，明确岗位责任，将贷款的调查岗、审查岗、审批岗、监督发放岗严格分离，各司其职、各负其责。

2）规范市场经济秩序，完善社会信用体系，为银行提供良好的信用环境。建立国民信用体系，建设全社会共享的信用数据库网络，建立企业的信用档案，将其银行贷款、信用状况、偿债能力等进行纪录，实施相应等级管理。对应个人贷款来说，建立个人信用记录，银行根据个人信用状况来决定是否发放贷款。完善信用评级指标体系，使银行可以对风险进行量化管理、对贷款进行精确定价，为银行防范信贷风险创造了条件。

（3）数据挖掘的数据准备

数据准备是根据目标定位提取信贷风险数据仓库中的数据，为数据挖掘风险分析做准备。确定目标数据，对数据预处理、数据缩放、数据过滤等，提取出符合目标定位的有效分析数据到目标数据表。ETL 过程实现数据的抽取、转换、装载到目标数据表。

信贷风险案例：数据准备

数据仓库中信贷风险表结构与说明如表 10-11 所示：

表 10-11　信贷风险表结构与说明

字　段　名	字 段 类 型	字 段 描 述
GOOD_BAD_CUSTER	INT	GOOD_BAD_CUSTER =1 代表申请贷款客户违约或严重违约 GOOD_BAD_CUSTER =0 代表申请贷款客户按时偿还贷款
APPL_ED	NUMBER(8)	客户申请贷款额度
CURR_DQED	NUMBER(8)	客户目前抵押贷款到期额度
CURR_DYGJ	NUMBER(8)	客户当前抵押财产估价
DKYY_TYPE	VARCHAR2(30)	DKYY_TYPE =DebtCon 表示合并的债务 DKYY_TYPE =HomeImp 表示家庭改善债务
JOB_TYPE	VARCHAR2(30)	客户职业类别
WORK_YEAR	INT	客户工作年数
ZYBS	NUMBER(3)	根据实际抵押物主要贬损数
TQBANK_ED	NUMBER(9,2)	客户拖欠银行额度
TQ_MONTHS	INT	客户拖欠贷款的月数
BANK_DC_TIMES	INT	银行催收次数
GIVECUST_MAXED	NUMBER(9)	银行给予客户贷款最高年限
LOAN_RATE	NUMBER(8,4)	银行债务收益比率

登录数据仓库，执行“select * from xdrish”语句，可以看到信贷风险表 xdrish 部分数据显示如图 10-12 所示。

GOOD_BAD_CUSTER	APPL_ED	CURR_DQED	CURR_DYGJ	DKYY_TYPE	JOB_TYPE	WORK_YEAR	ZYBS	TQBANK_ED	TQ_MONTHS
0	5000	47479	61690	DebtCon		5		0.00	65
0	5000	66839	87168	HomeImp	Other	4	0	0.00	123
0	5000	12457	53448	HomeImp	Self		0	0.00	207
0	5000	90059	113714	HomeImp	ProfExe	7	0	0.00	194
1	5000	22320	34900	HomeImp	Other	10	0	0.00	66
1	5000	40000	70500	HomeImp	Mgr	0	0	0.00	62
1	5000	100882	102700	HomeImp	Mgr	5	0	1.00	59
1	5000	31000	47350	HomeImp	ProfExe	12			
1	5000	63125	126500	HomeImp	Office	3	0	0.00	100
1	5000	100000	129694	HomeImp	Mgr	13			
1	5000		132000	HomeImp	Other	18	0	2.00	122
1	5000		9100	HomeImp	Other	1	0	0.00	55
1	5000	34200	50830	HomeImp	Other	6	0	0.00	57
1	5000		71277	HomeImp	Other	10	0	0.00	72
1	5000	123000	157500	HomeImp	Mgr	9			
1	5000	16600	25250	HomeImp	Other	6	0	0.00	89
1	5000		52000	HomeImp	Other		0	1.00	285
1	5000	22700	34225	HomeImp	Other	4	0	0.00	75
1	5000	6000	38575	HomeImp	Self	7	0	1.00	335

图 10-12　信贷风险表 xdrish 部分数据显示

【说明】 xdrish 表数据脚本存储在 qh 文件夹中，先把 xdrish.sql 数据脚本导入到数据仓库 Oracle 中。

对数据挖掘所需要的目标进行数据处理，根据需求提取信贷风险数据仓库中的数据，对

信贷风险历史数据表 xdrish 进行数据挖掘前的数据处理，生成数据挖掘需要的风险数据，其数据集通过 SAS 工具处理。信贷风险数据集 SAS 程序如下：

```
libname  qhsj  'd:\qh'; /*数据集存储文件夹逻辑库*/
libname  qhload  oracle user=chiran password=chiran path=orcl;
/*连接 Oracle 数据库的逻辑库*/
data  qhsj.creditrisk; /*生成信贷风险数据集*/
  set  qhload.xdrish;  /*读取数据仓库中的数据*/
run;
```

上面程序执行完成后生成信贷风险数据集 creditrisk，存储在 qhsj 逻辑库中。数据集信息变量分析解读，如表 10-12 所示。

表 10-12　creditrisk 数据集说明

变 量 名	模 型 角 色	度 量 方 式	描　述
GOOD_BAD_CUSTER	Target	binary	GOOD_BAD_CUSTER =1 代表申请贷款客户违约或严重违约 GOOD_BAD_CUSTER =0 代表申请贷款客户按时偿还贷款
APPL_ED	Input	Interval	客户申请贷款额度
CURR_DQED	Input	Interval	客户目前抵押贷款到期额度
CURR_DYGJ	Input	Interval	客户当前抵押财产估价
DKYY_TYPE	Input	Binary	DKYY_TYPE =DebtCon 表示合并的债 DKYY_TYPE =HomeImp 表示家庭改善债务
JOB_TYPE	Input	Nominal	客户职业类别
WORK_YEAR	Input	Interval	客户工作年数
ZYBS	Input	Interval	根据实际抵押物主要贬损数
TQBANK_ED	Input	Interval	客户拖欠银行额度
TQ_MONTHS	Input	Interval	客户拖欠贷款的月数
BANK_DC_TIMES	Input	Interval	银行催收次数
GIVECUST_MAXED	Input	Interval	银行给予客户贷款最高年限
LOAN_RATE	Input	Interval	银行债务收益比率

（4）模型的开发与算法的应用

数据挖掘过程中的关键点是模型的开发和算法的选择。模型的开发根据目标定位建立信贷风险模型。设计合理的数据挖掘模型是进行有效挖掘有用信息的关键环节。首先需要建立目标模型流程图，按照模型流程图进行数据挖掘。信贷风险模型流程如图 10-13 所示。

根据模型流程图和目标定位，选择合适的算法是挖掘有用信息的关键点，算法的选择要适合目标定位，根据响应变量和因变量之间的关系选择合适的算法。

信贷风险是对二值响应变量的判断，因此信贷风险选择逻辑回归算法进行数据挖掘。

本实验主要用到逻辑回归，既二值回归，也就是事件要么发生，要么不发生，其值用数值表示取值为 1 和 0 。运用的数学模型如下：

设事件的自变量为 x1，x2，x3，…，xk，某事件在自变量条件下发生的概率为 p，则此事件不发生的概率为 1–p。有时为更明显地看到事件的变化，可以求事件发生与不发生概率的比例，此实验的比例为发生事件概率除以不发生事件概率，用数学公式表示为 rate=p/(1–p)。

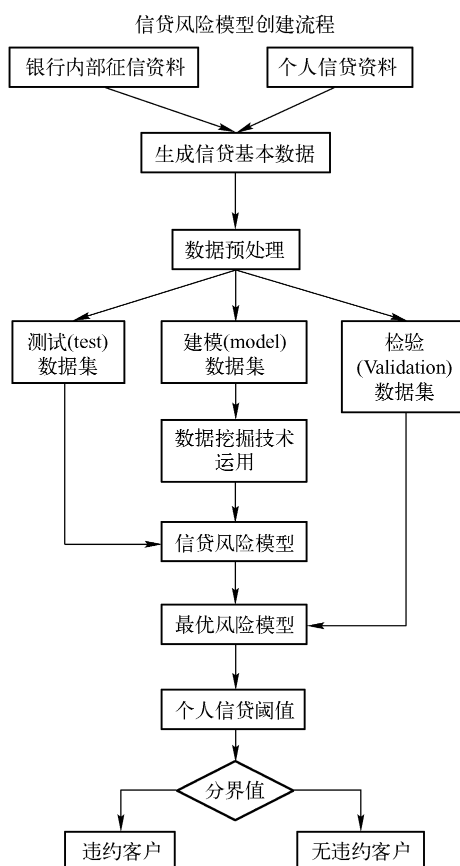


图 10-13　信贷风险数据挖掘模型流程

10.4　SAS/EM数据挖掘实现过程

根据业务需求和目标定位，处理数据挖掘数据，选择挖掘算法，建立模型。数据挖掘受两个重要因素的影响：一个是数据挖掘的数据质量和数据量大小，另一个是所选择的数据挖掘技术的合理性。如果数据质量和数据量达不到一定的程度，则挖掘出来的可能是错误的信息。数据挖掘技术的选择不合理可能对数据挖掘结果的精确度有很大的影响。

数据挖掘过程需要以下人员配合才能保证数据挖掘的成功。

1）业务人员：精通业务，理解并解释业务目标，根据业务需求确定业务数据和数据挖掘算法的业务需求。

2）数据分析人员：精通数据分析技术，能够把目标需求转化为数据挖掘的技术来实现。

3）数据库与数据仓库人员：精通数据库技术和数据仓库设计。

数据挖掘是一个多种专业人员合作开发的过程。目前世界上主流的数据挖掘工具有SAS/EM 数据挖掘模式、SPSS 的 Clementine 和 KXEN 公司的 KXEN 自动数据挖掘工具。具体应用哪个工具根据需求选择，但数据挖掘的流程和设计思路是不变的。本书使用 SAS/EM 进行数据挖掘。

信贷风险案例选择 SAS/EM 模块实现整个数据挖掘流程。根据 SAS/EM 的 SEMMA 流程实现信贷风险模型开发与应用，操作步骤如下。

（1）启动 SAS/EM

启动 SAS/EM 有以下两种方式。

① 在 SAS 命令窗口输入“miner”。

② 选择 SAS 菜单中的“解决方案”→“分析”→“企业数据挖掘”命令，打开数据挖掘模块，如图 10-14 所示。

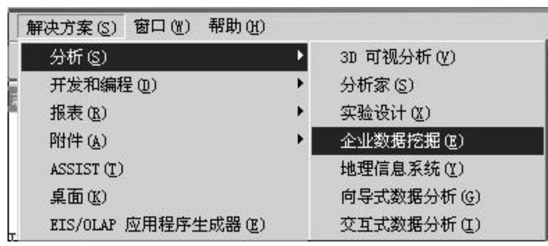


图 10-14　启动数据挖掘 EM 窗口

（2）创建数据挖掘工程

建立工程是为了便于管理某个数据挖掘的工程。所有与此工程相关的数据挖掘控件和流程图都由工程来控制和管理。

选择 SAS 菜单中的“文件”→“新建”→“项目”命令，在打开的 Create new Project 对话框中的 Name 文本框中输入 credit_risk 在 Location（存储工程的位置）处选择存储工程的文件夹，此数据挖掘实例工程存储在 D:\qh 文件夹里，如图 10-15 所示。

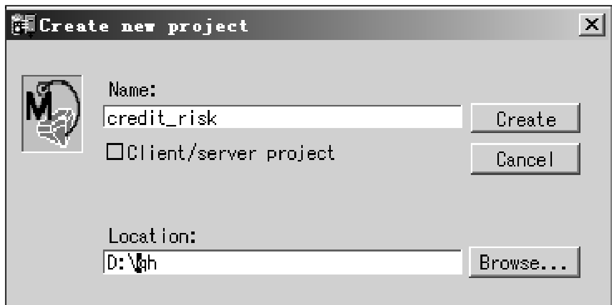


图 10-15　创建工程对话框

（3）获取数据源

添加一个输入数据源结点，为本次数据挖掘建立数据源。

将 Sample（抽样）控件的子控件节点 Input Data Source 拖动到流程图工作区，方法如下：

1）选择 Sample 控件下的 Input Data Source 结点，按住鼠标左键将 Input Data Source 拖入右侧的工作空间。

2）双击工作空间中的 Input Data Source 结点，进入 Input Data Source 的设置窗口。

3）在 Input Data Source 设置窗口中的 Data 项中单击 Select 按钮，选择数据源，进入 SAS

Data Set 窗口。在 Library 项选择 qhsj 逻辑库中的 creditrisk（信贷风险）数据集，如图 10-16 所示。

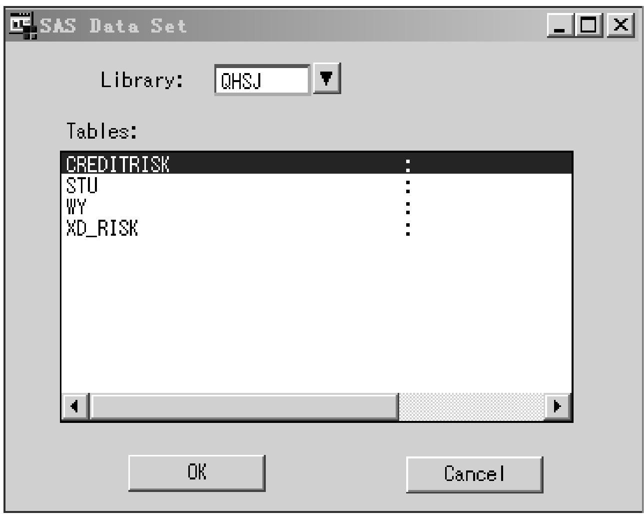


图 10-16　获取数据源窗口

4）单击“OK”按钮，出现如图 10-17 所示的窗口，Source Data 项为读入数据源项，通过 Select 按钮浏览逻辑库下的数据集。此实例对应的数据源为 QHSJ 逻辑库下的 CREDITRISK 数据集。Role 项可以取 RAW、TRAN、VALIDATE、TEST 和 SCORE。Metadata Sample（元数据样本）项显示出样本大小，默认为 2000。如果小于 2000，取元数据样本实际大小。此实例元数据样本为 5960。Columns 项显示出此元数据样本的变量个数，此实例变量为 13 个。通过 Change 按钮可以改动元数据样本大小。

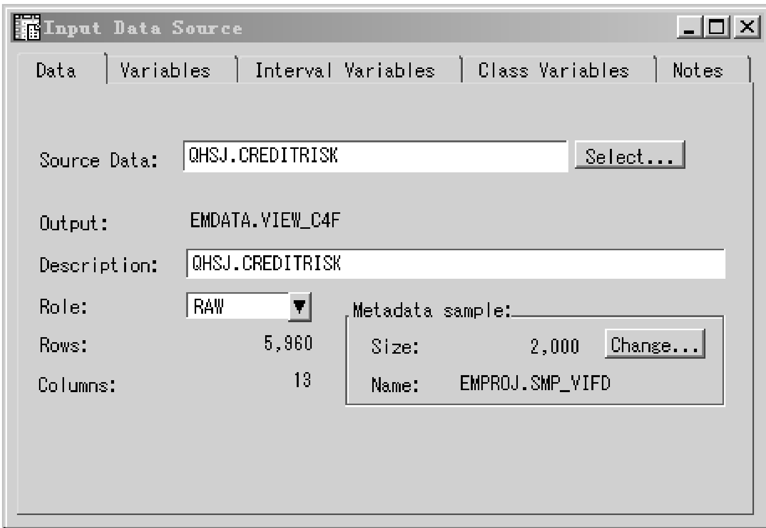


图 10-17　数据源信息窗口

选择 Input Data Source 窗口中的 Variables 可以浏览数据源变量，如图 10-18 所示。

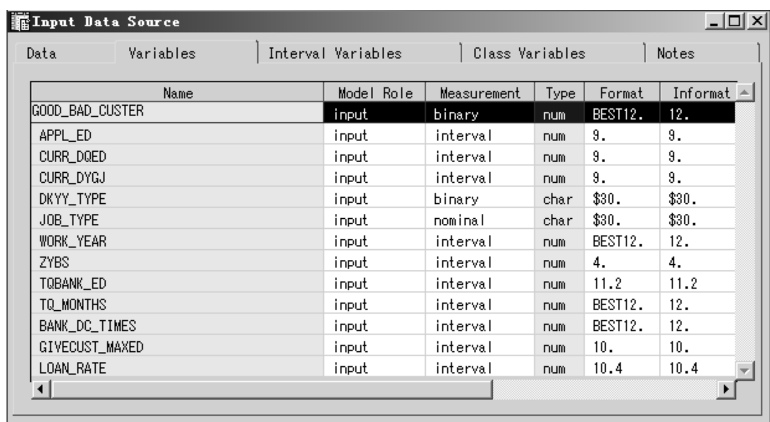


图 10-18　获取数据源窗口 Variables 选项对应变量窗口

选择 Input Data Source 窗口中的 Interval Variabels 可以浏览数据源区间变量，选择 Class Variables 可以浏览查看变量的分类情况。Notes 项可以添加注释，写一些便于阅读此数据挖掘的信息，相当于程序中的注释语句。此实验通过 Variables 项找到 GOOD_BAD_CUSTER 数据项，在 GOOD_BAD_CUSTER 行的 Model Role 项处单击鼠标右键，在弹出的快捷菜单中选择 Set Model Role（设置模型角色），然后在弹出的菜单中选择 target，如图 10-19 所示。

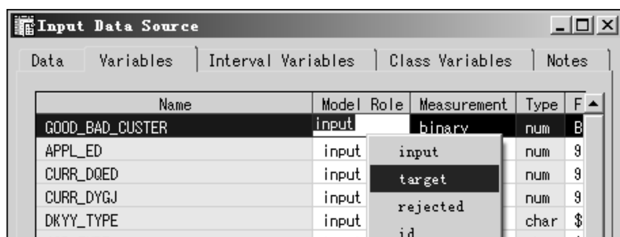


图 10-19　模型角色设置

设置好数据挖掘目标变量后，数据挖掘分析的目标变量属性显示如图 10-20 所示。

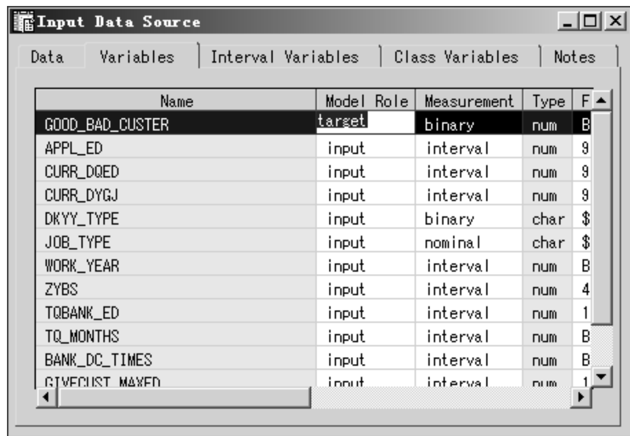


图 10-20　GOOD_BAD_CUSTER 变量设置

5）关闭 Input Data Source 设置窗口，保存设置，显示数据源控件 QHSJ.CREDITRISK，如图 10-21 所示。

6）数据源获取完成后，要运行对控件，使数据源生效，在数据源控件 QHSJ. CREDITRISK 处单击鼠标右键，在弹出的快捷菜单中选择 Run，如图 10-22 所示。



图 10-21　数据源控件 QHSJ.CREDITRISK

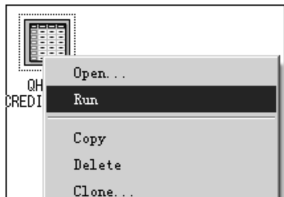


图 10-22　运行数据源控件

（4）数据源探测

为对获取到的数据源有一个整体的认识，需要用数据源探测控件对变量分布进行一个探测，以掌握哪些变量对实现数据挖掘目标分析更重要。

将 Explore（探测）控件的子控件节点 Insight 拖动到流程图工作区，然后把鼠标放到 QHSJ.CREDITRISK 数据源上，当光标变成“+”形状时按住鼠标左键画线到 Insight 节点，这样就建立了数据源到探测节点的连接，如图 10-23 所示。双击 Insight 节点打开 Insight 窗口，如图 10-24 所示。

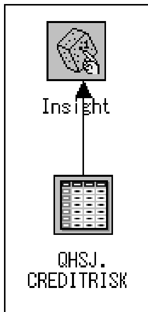


图 10-23　QHSJ.CREDITRISK TO Insight 窗口

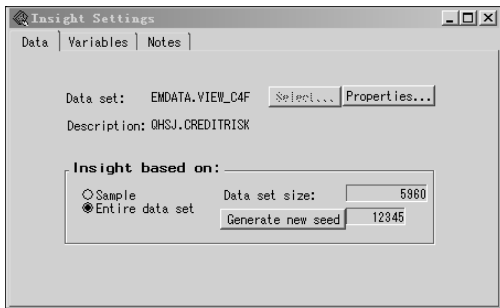


图 10-24　Insight 窗口

通过 Insight 窗口看到探测节点可以基于的探测方式为 Sample（抽样数据集默认样本大小）和 Entire data set（实际的数据集大小），Variables 标签浏览变量属性特征，NOTES 添加注释用的说明文字。

此实例选择基于探测方式为 Entire data set（实际的数据集大小），关闭此窗口。在出现的询问是否保存此窗口的设置中选择“是”，保存设置。然后单击鼠标右键选择“Run”，在出现的询问窗口是否浏览结果中选择“是”，出现的结果如图 10-25 所示。

运行此探测节点。运行完后，选择菜单栏的“分析”，然后选择“分布”，在出现的窗口按住〈Ctrl〉键可以选择要查看的变量，然后单击“Y”按钮就可以了。此实例只选择 GOOD_BAD_CUSTER 变量查看变量分布，可以看出变量 GOOD_BAD_CUSTER 对应值为 1 的记录为 1189 个，结果显示如图 10-26 所示。

（5）对分析数据集进行拆分

为验证模型，此实验把源数据集拆分成训练数据集（Train）（占源数据集的 60%）、校验

数据集（Validation）（占源数据集的 20%）和测试数据集（Test）（占源数据集的 20%）。

EMDATA.VIEW_C4F

13	区间型	区间型	区间型	区间型	
5960	GOOD_BAD_CUSTER	APPL_ED	CURR_DQED	CURR_DYGJ	
1	0	5000	47479	61690	DebtCon
2	0	5000	66839	87168	HomeImp
3	0	5000	12457	53448	HomeImp
4	0	5000	90059	113714	HomeImp
5	1	5000	22320	34900	HomeImp
6	1	5000	40000	70500	HomeImp
7	1	5000	100882	102700	HomeImp
8	1	5000	31000	47350	HomeImp
9	1	5000	63125	126500	HomeImp
10	1	5000	100000	129694	HomeImp
11	1	5000	.	132000	HomeImp
12	1	5000	.	9100	HomeImp
13	1	5000	34200	50830	HomeImp
14	1	5000	.	71277	HomeImp
15	1	5000	123000	157500	HomeImp
16	1	5000	16600	25250	HomeImp
17	1	5000	.	52000	HomeImp
18	1	5000	22700	34225	HomeImp
19	1	5000	6000	38575	HomeImp
20	1	5000	.	74500	HomeImp
21	1	5000	.	94113	HomeImp
22	0	5000	73126	87500	DebtCon

图 10-25　Insight 运行结果浏览窗口

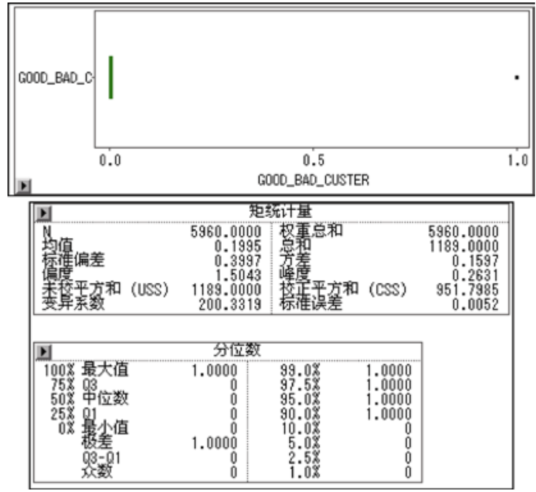


图 10-26　GOOD_BAD_CUSTE R 变量探测分布信息

将 Sample（抽样）控件的子控件节点 Data Pattition 拖动到流程图工作区，然后把鼠标放到 DZ.CREDIT 数据源上，当光标变成“+”形状时按住鼠标左键画线到 Data Partition 节点，这样就建立了数据源到数据拆分节点的连接，如图 10-27 所示。

双击“Data Partiton”节点打开 Data Partiton 窗口，如图 10-28 所示。

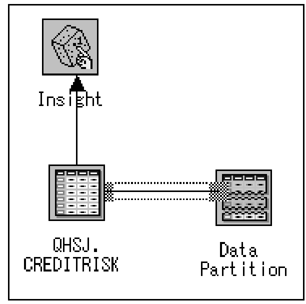


图 10-27　Data Partition 节点窗口

Data Partition
Notes
Data | Variables | Partition | Stratification | User Defined | Output
Method:
Simple Random
Stratified
User Defined
Random Seed:
Generate New Seed　12345
Percentages:
Train: 60 %
Validation: 20 %
Test: 20 %
Total: 100 %

图 10-28　打开 Data Partition 节点显示窗口

选择 Partition 标签的窗口进行数据集拆分，在 Method（抽样方式）选项区选择 Simple Radom（随机抽样）；在 Percentages（数据集拆分占百分比）选项区把 Train（训练集）修改为占总数据集的 60%，Validation（校验集）修改为占总数据集的 20%，Test（测试集）修改为占总数据集的 20%。通过数据集的拆分，可以得出在本实验，训练集记录条数为 5960×0.6=3576，校验集记录条数为 5960×0.2=1192，测试集记录条数为 5960×0.2=1192。设置完成后进行保存并关闭该窗口。

（6）替换数据集中的缺失值

由于回归和神经网络模型不具有对缺失值的处理功能，此处用处理缺失值接点处理缺失值。replacement 用来处理数据源中的缺失值。

实际开发中对于回归模型或神经网络模型都需要 replacement 控件处理分析数据源的缺失值。

将 Modify（修改）控件的子控件节点“Replacement”拖动到流程图工作区，然后把鼠标放到 Data Partition 节点上，当光标变成“+”形状时按住鼠标左键画线到 Replacement 节点，这样就建立了属性设置节点到替换缺失值节点的连接，如图 10-29 所示。双击 Replacement 节点打开 Replacement 窗口，如图 10-30 所示。

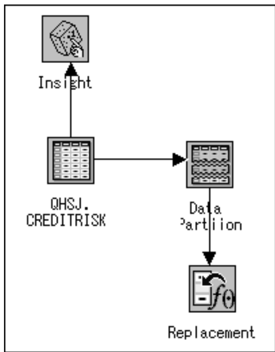


图 10-29　Replacement 节点窗口

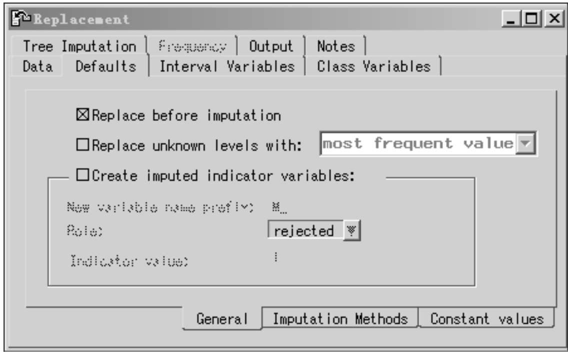


图 10-30　打开 Replacement 显示窗口

本案例选择 Repalce before imputation，在输入前替换缺失值。

通过此窗口可以进行缺失值设置，如选择“Imputaiton Methods”可以设置替换缺失值的方式，如图 10-31 所示。

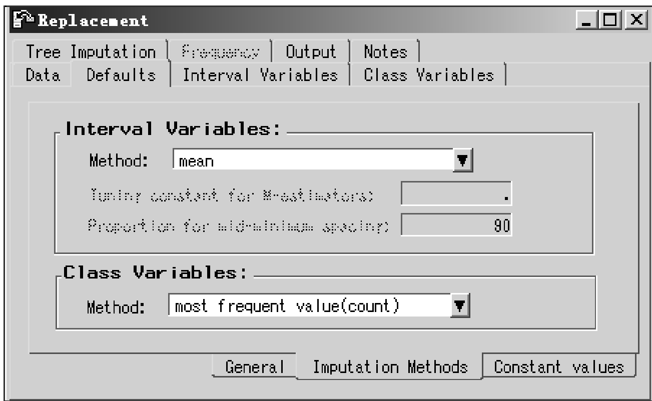


图 10-31　打开 Replacement 中的 Imputation Mthods 标签显示窗口

通过此图看到 Interval Variables 类型的变量替换方式可以在 Method 项单击向下的小箭头进行选择，如选择 mean（均值替换）；Class Variables 类型的变量替换方式可以在 Method 项单击向下的小箭头进行选择，如用 most frequent value（count）（最大频率值替换）。

【说明】 Interval 类型的变量用抽样数据集的均值替换缺失值。

Normal、Binary 和 Ordinal 类型的变量用抽样数据集中的最大频率值替换。

（7）建立模型

二分类 Logistic 回归模型适合于目标变量为二项分类，其目标变量值只有两个分类值，如

1 和 0。在本实验中 GOOD_BAD_CUSTER 可以取的值为 1 和 0 两个值，GOOD_BAD_CUSTER 为“1”时，表示申请贷款客户违约或严重违约；GOOD_BAD_CUSTER 为“0”时，表示申请贷款客户按时偿还贷款。

将 Model（模型）控件的子控件节点 Regression 拖动到流程图工作区，然后把鼠标放到 Replacement 节点上，当光标变成“+”形状时按住鼠标左键画线到 Regression 节点，这样就建立了缺失值替换节点到回归模型节点的连接，如图 10-32 所示。

双击 Regression 节点，打开回归模型，Variables 项显示信息如图 10-33 所示。

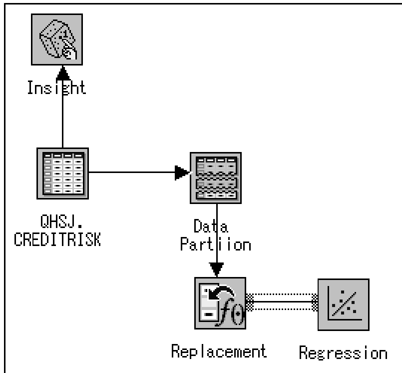


图 10-32　Regression 节点

Name	Status	Model Role	Measurement	Type	Format
GOOD_BAD_CUSTER	use	target	binary	num	BEST12.
APPL_ED	use	input	interval	num	9.
CURR_DQED	use	input	interval	num	9.
CURR_DYGJ	use	input	interval	num	9.
DKYY_TYPE	use	input	binary	char	$30.
JOB_TYPE	use	input	nominal	char	$30.
WORK_YEAR	use	input	interval	num	BEST12.
ZYBS	use	input	interval	num	4.

图 10-33　Regression 模型 Variables 项显示信息

选择图 10-33 中的 Model Options 项可以对模型类型 Tyep 进行设置。本案例选择 Logistic（逻辑回归模型），回归函数 Link Funciton 为 LOGIT，如图 10-34 所示。

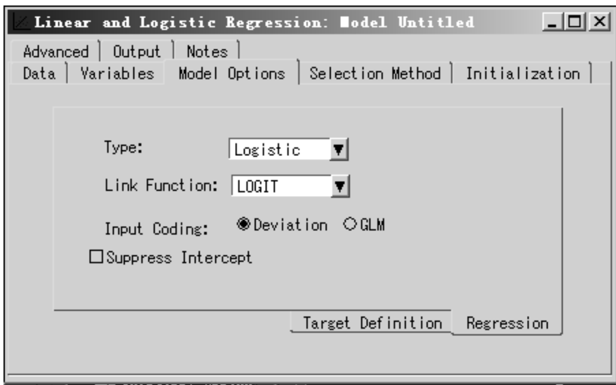


图 10-34　Model Options 选项设置

【提示】 回归模型在使用前需要对读入的数据源 Input Data Source 控件先指定目标变量（target）。

【注意】 回归模型适合的目标变量为如下类型的目标变量的属性：Interval、Ordinal、Nominal 和 Binary 类型。

选择图 10-34 中的 Selection Method 标签，可以设置回归方式，如图 10-35 所示。

此案例选择 Stepwise 回归方式。由于不考虑数据集中两个或两个以上变量联合作用于目标变量，不需要对 Effect Hierarchy 进行设置。

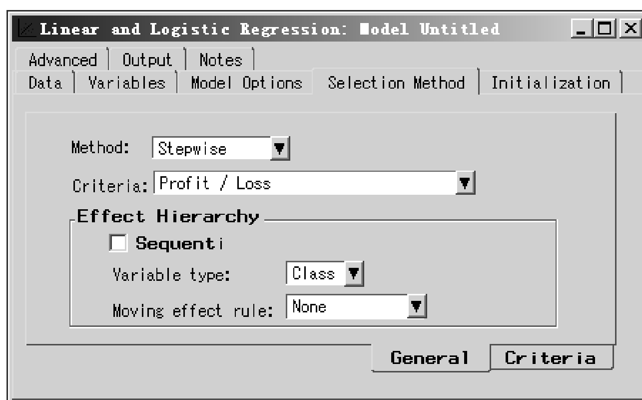


图 10-35　Linear and Logistic Regression:Model 中的 Selection Method 标签窗口

【说明】 常用的回归模型有以下 3 种方式：

1）FORWARD：前进法一开始模型中没有变量，每次将一个最显著的变量引入模型，直到模型以外的变量不再有显著的值为止。

2）BACKWAND：后退法一开始模型中含所有自变量，每次从模型中剔除一个贡献最小的变量，直到模型中只剩下均为显著的变量为止。

3）STEPWISE：逐步法每次引入模型一个最显著的变量，然后考虑从模型中剔除一个最不显著的变量，直到既没有变量引入也没有变量剔除为止。

关闭此窗口，在出现的对话框中保存设置，如图 10-36 所示。

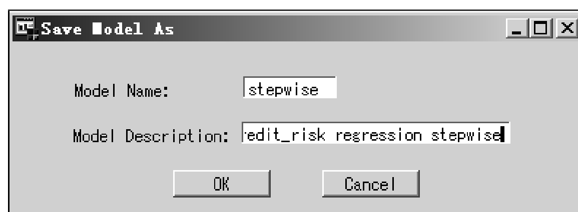


图 10-36　Save Model As 对话框

① Model Name（模型名字）：对应项输入模型名字，本案例输入 stepwise。

② Model Description（模型描述项）：输入对模型的描述信息。

至此整个回归模型已经建立完成，单击 Regression 节点，运行回归模型，在出现的对话框中单击“OK”按钮可以查看运行结果。选择回归模型运行结果的“Output”标签，如图 10-37 所示。

通过图 10-37 可以看到训练集有 3576 个信贷客户，其中违约信贷客户为 704，无违约信贷客户 2872 信贷客户。

选择 Estimates 可以看参数评估信息，可以查看显示回归模型中有效的 Effect T-score。对输入变量而言，输入变量的重要性的预测从左到右逐渐减小，表示输入变量对模型的影响度逐渐减小，也就是变量对模型的影响左边的 TQBANK_ED 变量最重要，其次为 TQ_MONTHS，其他依次为 ZYBS、LOAN_RATE 和 BANK_DC_TIMES。对影响目标变量重要的输入变量加以防范可以减少违约信贷客户，如图 10-38 所示。

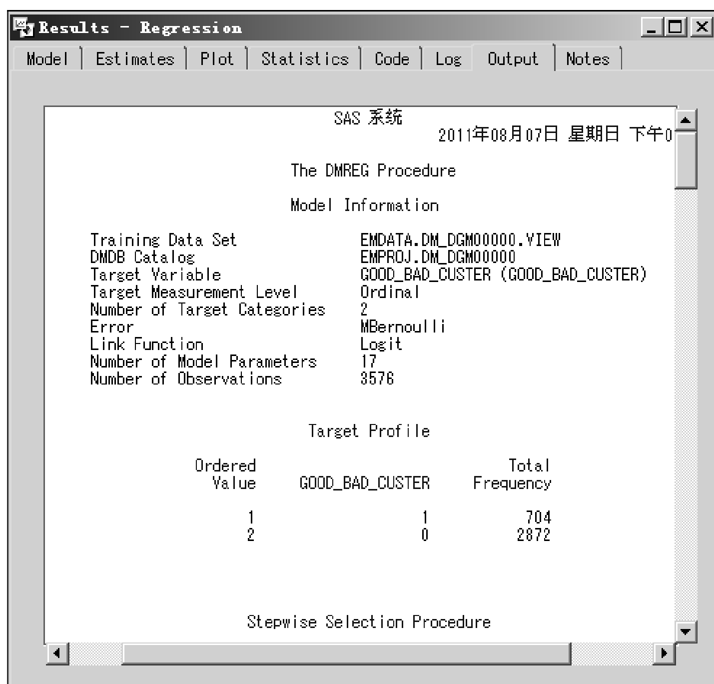


图 10-37 Results- Regression 显示信息

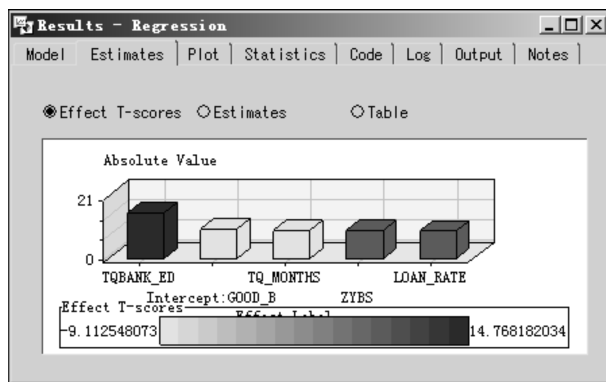


图 10-38 Estimate 项显示信息

在回归模型上单击鼠标右键，在弹出的对话框中选择“Model Manger”（模型管理），在模型管理中选择“工具”，在弹出的菜单中选择“升降图”，如图 10-39 所示。

（8）建立决策树模型

对于决策数模型，不需要替换缺失值。对于数据量比较大的数据集，决策树有可能失去作用。决策树也经常用来对二值变量进行分类。

将 Model（模型）控件的子控件节点 Tree 拖动到流程图工作区，然后把鼠标放到 Data Set Partiion 节点上，当光标变成“+”形状时按住鼠标左键画线到 Tree 节点，这样就建立了数据属性设置节点到决策树模型节点的连接，如图 10-40 所示。

双击 Tree 节点打开 Tree 窗口，Variables 项显示信息，如图 10-41 所示。

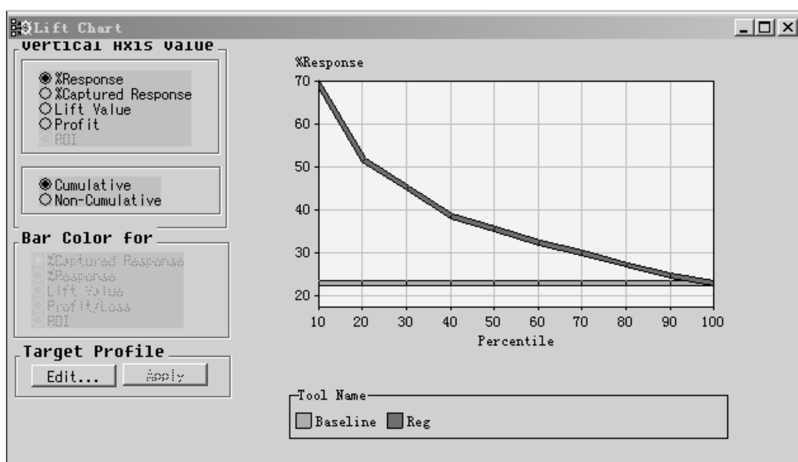


图 10-39　选择“升降图”

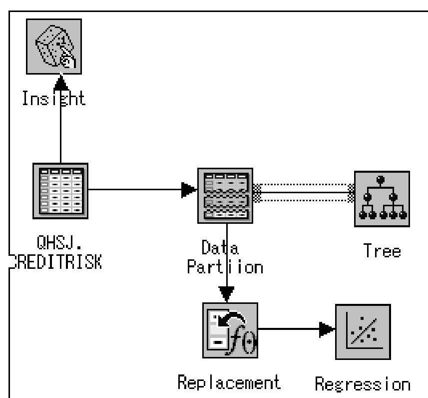


图 10-40　决策树 Tree 节点

Tree: Model Untitled

Data | Variables | Basic | Advanced | Score | Notes

Name	Status	Model Role	Measurement	Type	Format
GOOD_BAD_CUSTER	use	target	binary	num	BEST12.
APPL_ED	use	input	interval	num	9.
CURR_DQED	use	input	interval	num	9.
CURR_DYGJ	use	input	interval	num	9.
DKYY_TYPE	use	input	binary	char	$30.
JOB_TYPE	use	input	nominal	char	$30.
WORK_YEAR	use	input	interval	num	BEST12.
ZYBS	use	input	interval	num	4.
TQBANK_ED	use	input	interval	num	11.2
TQ_MONTHS	use	input	interval	num	BEST12.

图 10-41　Tree 模型 Variables 项信息显示

单击 Basic 标签可以查看决策树的基本信息，如图 10-42 所示。

Tree: Model Untitled

Data | Variables | Basic | Advanced | Score | Notes

Splitting criterion
- ◉ Chi-square test
- ○ Entropy reduction
- ○ Gini reduction

Significance level: 0.200

Minimum number of observations in a leaf:	5
Observations required for a split search:	35
Maximum number of branches from a node:	2
Maximum depth of tree:	6
Splitting rules saved in each node:	5
Surrogate rules saved in each node:	0

☒ Treat missing as an acceptable value

图 10-42　Basic 项信息显示

此决策树的Basic标签窗口显示了默认值，卡方检验的显著水平默认值为0.200；“Minmum number of observations in a leaf”项对应的是决策树叶子节点最小的观测值 5；“Maximum number of branches from a node”项对应的是决策树一个节点的分支最大分支数2；“Maximum depth of tree”项是设置决策树的深度，此处为6，表示最大树高为6层。对于上面的默认设置可以根据需求进行更改，此案例选择默认值。

对 Basic 标签项属性的解释说明如下：

1）Splitting criterion。

划分标准项选项说明：根据目标变量的度量方式进行选择，对于定性或二进制的目标变量，可以选择以下划分规则。

① Chi-sqare test：x^2检验（默认）——Pearson x^2检验用于衡量对目标变量建立分支结点，其默认Significance level（显著性指标）为0.20。

② Entropy reduction：熵值约简，通过对熵值大小的衡量反映结点不纯性，也称为熵不纯性。

③ Gini reduction：基尼系数约简，通过对基尼系数大小的衡量反映结点不纯性，也成为Gini 不纯性。

2）对于连续型目标变量，可以选择以下两种划分规则：

F 检验（默认）——F 检验的 P 值与结点一致性有关，默认显著性指标为 0.20。

一致性约简——该约简基于结点的均方差检验。

本案例中目标变量家庭资产抵押贷款为二元变量，选择Chi-square test（x^2检验）是一个比较合适的划分规则。

同时可以在 Basic 标签指定以下决策树相关属性值。

Minimum number of observations in a leaf：指定叶节点包含的最小观察记录（default = 1）。

Observations required for a split search：指定划分的节点小于此处定义的条件，此处节点记录数就不再划分分支。该选项保证划分的结点都有观测数据，并且对于比该项指定的观测值还要少的结点不进行继续划分。

Maximum number of branches from a node：指定拆分节点分支的最大观察记录数（default= 2）。

Maximum depth of tree：指定树的最大高度（default = 6）。

Splitting rules saved in each node：指定每一个节点保存划分规则保留项（default = 5）。

Surrogate rules saved in each node：指定每一个节点的保存替代规则（default = 0）。

Treat missing as an acceptable value：对处理的缺失值替换。

决策树“Advanced”标签项可以对 Sub-tree（子树）进行高级设置，如图 10-43 所示。

Model assessment measuer：评估度量模型的选择。评估度量模型是基于从有效的数据中获得的决策树结果中选择出最佳的决策树模型，对于范畴型目标变量或间断型目标变量会有不同的评估度量模型选择。

Sub-tree：自定子树模型。该选项在于设定怎样判断一棵决策树已经足够大了。

P-value adjustment：p值调整方法的选择。如果既没有选择x^2检验也没有选择F检验，就需要指定一种方法调整p值。

单击鼠标右键，选择 Run 运行决策树模型。从决策树运行窗口中显示的信息可以看出Leaves对对应的值为3，表示此决策树的高度为3层，如图 10-44 所示。

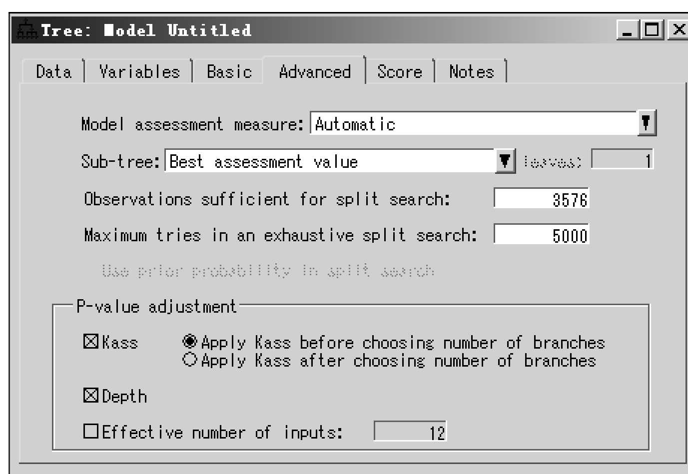


图 10-43　决策树 Variables 项设置显示

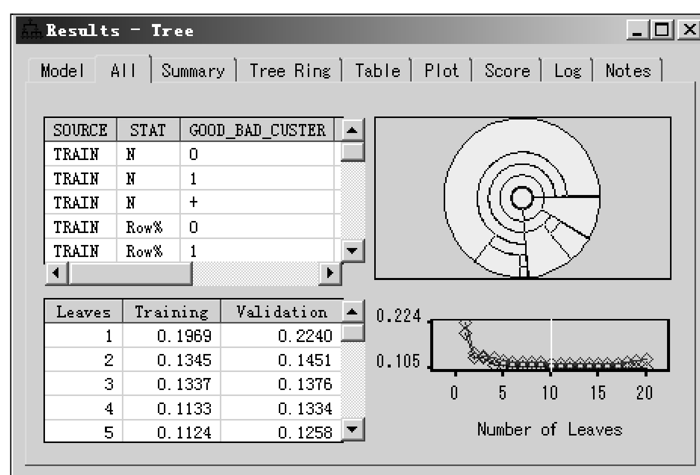


图 10-44　Results-Tree 窗口

单击 SAS 菜单的“查看”，在弹出的下拉菜单中选择“树状结构（T）”，可以看到决策树的高度和叶节点显示信息。此决策树的树状结构图显示了树的高度为 3 层，对于训练集的观测 Total 显示总观测记录为 3576 位信贷客户。Good_bad_custer 为 1 的叶节点显示违约信贷客户的训练数据集记录为 704 位客户；Good_bad_custer 为 0 的叶节点显示无违约信贷客户的训练数据集记录为 2872 位客户。决策树还显示了训练数据集的无违约信贷客户与违约信贷客户占的百分比，同时根据每一个变量的信息熵，也就是贡献度进行对树分类，如图 10-45 所示。

树图包含下列项目：

根节点——树中包含所有案件的顶部节点。

内部节点——非终结节点（也包括根节点），其中包含分裂规则。

叶节点——终端节点包含一组观察的最终分类。

可以使用滚动条来显示其他节点。预期损失值用于递归分区同质群体中的数据。

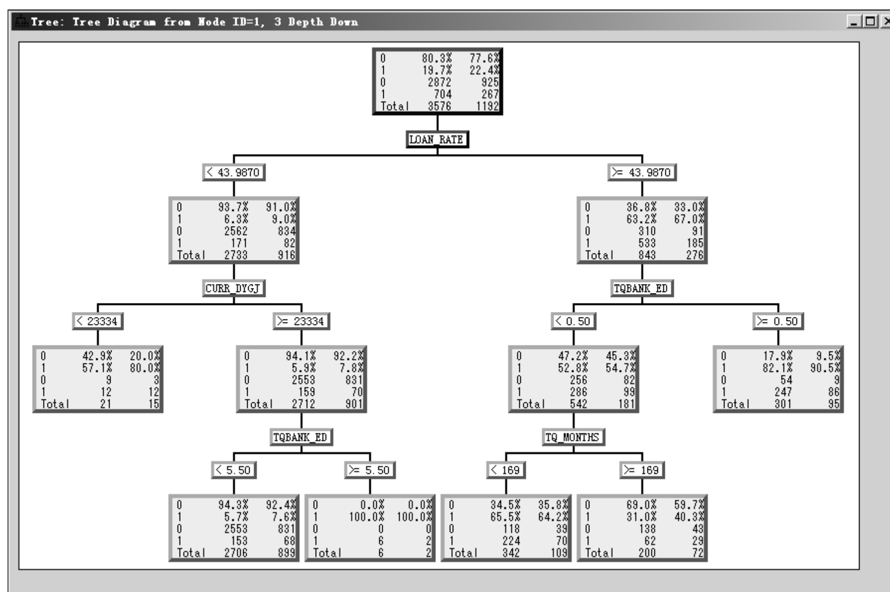


图 10-45　树分类显示信息

当使用的损失评估标准建立树，每一个节点行包含下列数据：

第一行列出了在训练数据集和检验数据集的无违约信贷客户的百分比。

第二行列出了在训练数据集和检验数据集的违约信贷客户百分比。

第三行列出了训练数据集和检验数据集的无违约信贷客户的数目

第四行列出了训练数据集和检验数据集的违约信贷客户的数目。

第五行列出了训练数据集和检验数据集的信贷客户各自对应的总观测数。

关闭决策树运行显示的信息窗口，下面将要进行模型评估。

（9）对回归模型和决策树模型建立模型评估

为检验两个模型哪个模型最优，可以通过 SAS/EM 的模型评估控件进行比对。

1）将 Assess（评价）控件的子控件节点 Assessment 拖动到流程图工作区，把控件 Assessment 放到 Tree 节点下面，然后分别把鼠标放到 Tree 节点和 Regression 上，当光标变成“+”形状时按住鼠标左键画线到“Assessment”节点，这样就建立了决策树节点和回归节点到模型评价节点的连接，如图 10-46 所示。

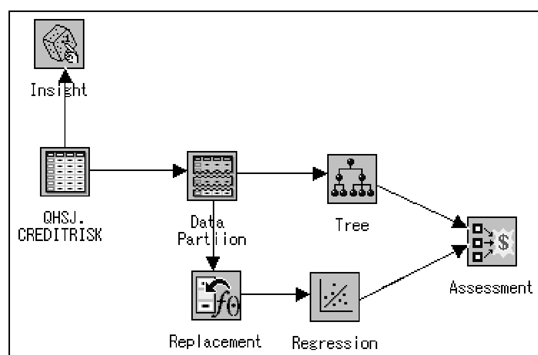


图 10-46　评价节点 Assessment

2）双击 Assessment 节点打开 Assessment Tool 窗口，此窗口显示评价的两个模型为 Regression 和 Tree，如图 10-47 所示。

Assessment Tool

Models | Options | Reports | Output

Tool	Name	Description	Target	Target Event
Regression	stepwise	credit_risk regression stepwise	GOOD_BAD_CUSTER	1
Tree	Untitled	Tree	GOOD_BAD_CUSTER	1

图 10-47　Assessment Tool 窗口

3）单击鼠标右键，在弹出的窗口选择 Run 运行此评价控件。在弹出的窗口询问是否浏览结果，单击“是”按钮，在出现的窗口按住〈shift〉键，选择 Regression 和 Tree 两个模型，然后选择菜单栏上的“工具”，在弹出的窗口选择“升降图”，评价控件运行后的升降图显示了模型比对，Cuumulative 表示显示的为积累升降图。评价控件中的每一个模型都给信贷客户 Good_bad_custer 打一个分值，分数从高到低排列，分为 10 份。纵轴表示响应率，横轴表示分值。对于回归模型，信贷客户在前 10%的银行对信贷客户违约响应率为 70%左右；对于决策树模型，信贷客户在前 10%的银行对信贷客户违约响应率为 90%。但当信贷客户在 90%时，可以看出回归模型对二值响应率更高，如图 10-48 所示。

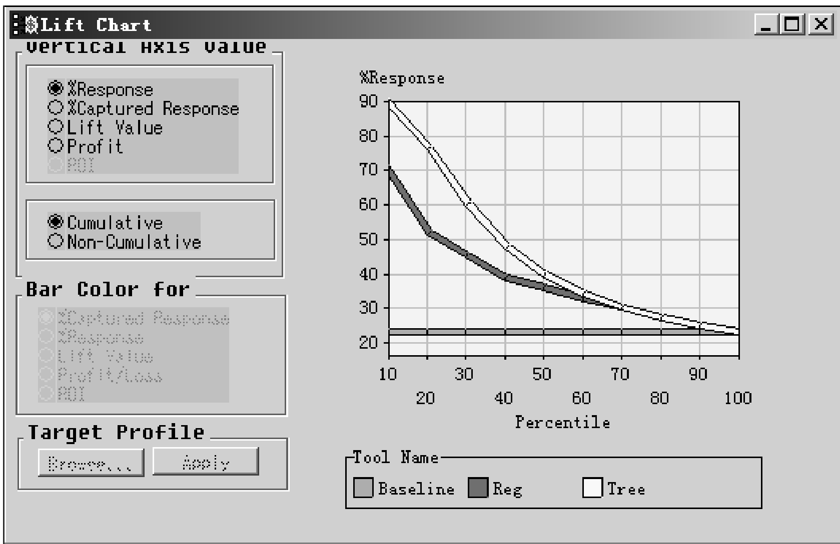


图 10-48　Lift Chart 窗口

（10）模型应用

为把模型应用到原始数据集，这里引入打分模型，然后通过打分模型把模型应用到所有数据集。

1）将 Scoring（打分）控件的子控件节点 Score 拖动到流程图工作区，把控件放到 Assessment 节点下面，然后把鼠标放到 Assessment 节点上，当光标变成“+”形时按住鼠标左键画线到 Score 节点，这样就建立了评价节点和打分节点的连接，如图 10-49 所示。

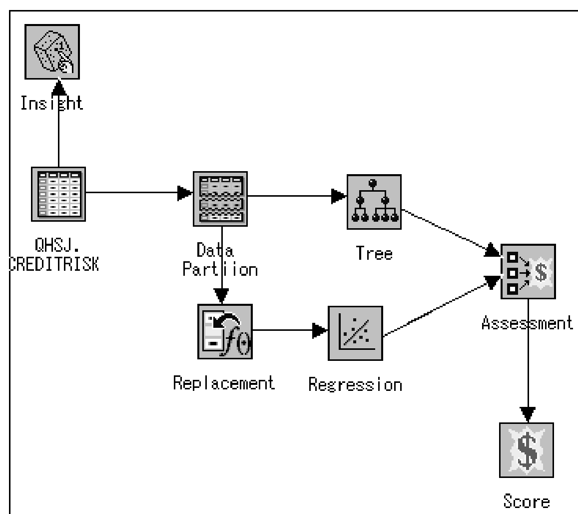


图 10-49　socre 节点

双击 Score 节点打开 Score 窗口，本实验选中 Apply training data score code to score data set 单选按钮，把训练数据得分应用到源数据集，如图 10-50 所示。

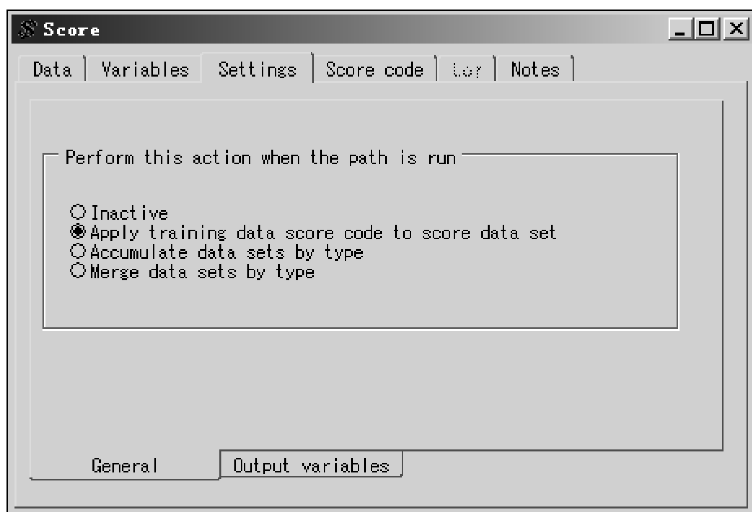


图 10-50　Score 选项设置

2）引入源数据集。

源数据集引入到打分节点，通过打分节点引入的模型应用到源数据集所有数据。

将 Sample（抽样）控件的子控件节点 Input Data Source 拖动到流程图工作区，把控件放到 Score 节点左侧，然后把鼠标放到 Input Data Source 节点上，当光标变成“+”形时按住鼠标左键画线到 Score 节点，这样就建立了输入数据源节点到打分节点的连接，如图 10-51 所示。

双击 Input Data Source 节点打开 Input Data Source 窗口，对 Input Data Source 窗口中的 Role 进行设置，从下拉列表中选择 SCORE，打分应用源数据，如图 10-52 所示。

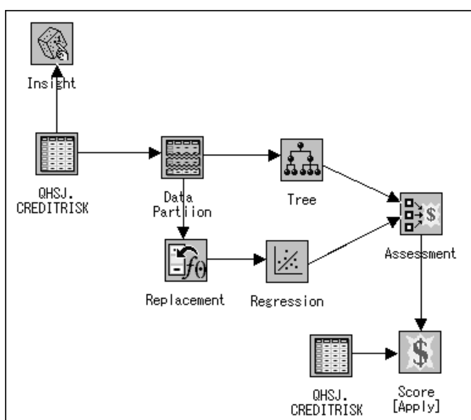


图 10-51　引入源数据集到模型应用

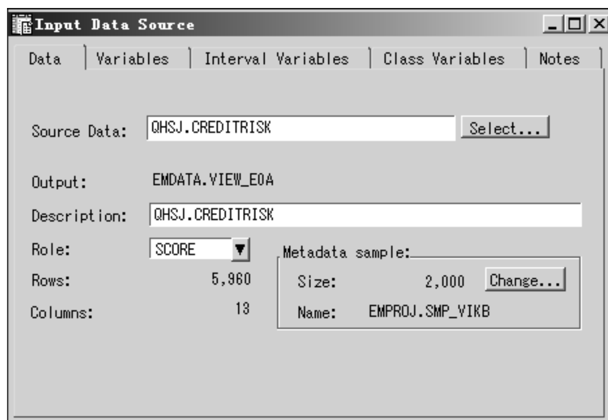


图 10-52　Input Data Source 节点设置

（11）信贷客户分布查看

对建立好的模型通过 Explore 中的 Distribution Explorer 子节点可以查看结果。

将 Explore（探测）控件的子控件节点 Distribution Explorer 拖动到流程图工作区，把控件放到 Score 节点下面，然后把鼠标放到 Score 节点上，当光标变成“+”形状时按住鼠标左键画线到 Distribution Explorer 节点，这样就建立了打分节点到探测节点的连接，如图 10-53 所示。

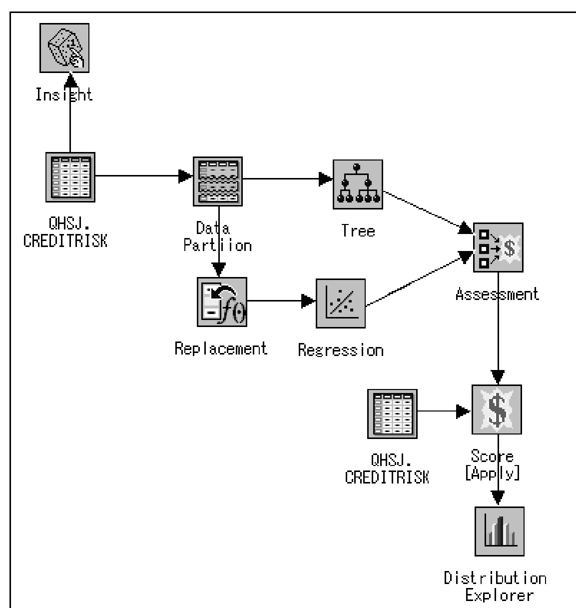


图 10-53　Distribution Explorer 节点

【注意】 在运行 Distribution Explorer 节点前先运行 Score 节点。

单击鼠标右键，选择 Run 运行，在弹出的询问是否查看结果窗口中单击“是”按钮，选择 SAS 菜单中的“查看”，在弹出的子菜单中选择“透视”，然后在弹出的菜单中选择“3D 视图”，此图显示了因变量 Good_bad_custer 的百分比，违约信贷客户占比为 19.3，结果显示如图 10-54 所示。

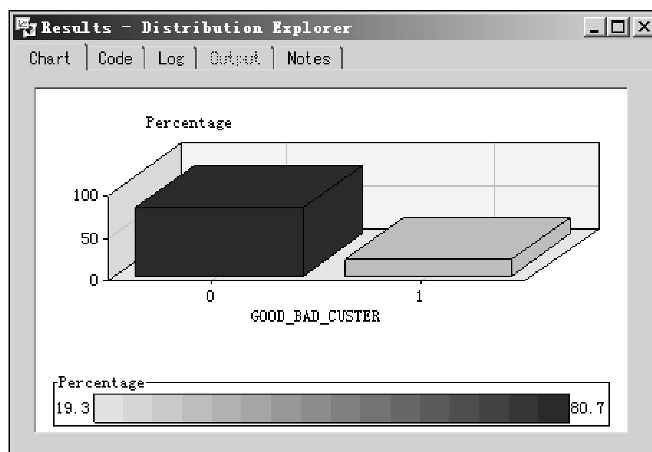


图 10-54　透视图

通过运行 SAS/EM 工具，查看运行报告，进行分析，得出如下评价和分析：

本实验通过回归模型中的逻辑回归和决策树模型的应用挖掘出了信贷客户的违约客户和无违约客户，满足了信贷客户模型的需求。通过对比模型可以发现逻辑回归在二分类目标变量是最适合的，防止了过度拟合，同时将信贷客户根据评分总值和阈值进行比对，来判断哪个客户是违约信贷客户，哪个客户是无违约信贷客户。根据数据挖掘分析来控制某一个信贷变量或几个信贷变量，减少信贷违约客户，尽可能地降低信贷客户违约风险。

第 11 章　SAS模型开发案例

随着计算机和互联网技术的快速发展和广泛应用，我们的社会正在经历一场数字化和信息化的变革，各种信息采集终端的出现，使得信息的获取变得更加容易，从日常的数字和影像采集设备，到企业高速膨胀的数据仓库，数据的积累为基于数据的科学决策提供了可能性。同样，日益激烈的市场竞争，要求企业改变传统的基于经验的粗放简单的决策方式，代之以精确的客户关系管理方法，从而降低经营成本，保持竞争力。另外，伴随着社会全面的信息化过程，云计算及移动智能终端的广泛应用，势必带来数据的全面融合，引发新一轮智能化的浪潮。智能化作为一个数据信息应用的系统化过程，涉及数据信息的获取、共享、整合、分析、建模和分发等方面，而在这之中，数据模型的开发将会作为整个智能化的中枢，变得日益重要，如图 11-1 所示。

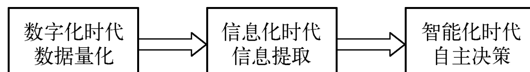


图 11-1　信息化发展流程

11.1　数据挖掘建模概述

11.1.1　数据挖掘层级

基于企业对数据的使用需求情况，数据挖掘层级可以分为以下 4 个层级，即报表级应用、分析级应用、模型级应用和智能决策系统。

（1）报表级应用

报表级应用的重点是回顾历史。它主要是利用数据对企业的经营历史进行回顾和总结，并且定期或不定期生成报表。通过这些报表可以回答某段时间内某个领域发生了什么事情，有多少次、多大数量等问题。同时这些报表兼具一定的查询功能。报表级应用可以进一步细分为常规报表（BI）和即席查询两部分。

（2）分析级应用

分析级应用的重点是关注现在。它主要是根据业务需要，对数据从不同维度进行展开和比较，可通过一些复杂的分析（频次分析、相关分析等）发现问题出现的原因，并可以根据业务需求设定一定的警戒值，起到提醒和警示作用。报表级应用可以进一步细分为多维分析（OLAP）、预警提醒和统计分析。

（3）模型级应用

模型级应用的重点是预测未来。它主要是基于历史数据，开发各种预测模型，对客户和业务的未来发展作出预测，进一步设计和优化策略方案，实现企业未来效益的最优化。模型

级应用可以进一步细分为预测建模和策略设计优化。

（4）智能决策系统

智能决策系统的重点是决策实施。它主要是指将基于各种预测模型得到的策略结果，及时准确地分发到业务前端供业务人员使用，同时收集并跟踪模型执行结果。智能决策系统可以进一步细分为策略部署和跟踪报表。

11.1.2 挖掘建模概念

数据挖掘建模就是通过模型开发的方法，从海量的数据中筛选出有用的信息和规律，以实现对目标的准确刻画。数据模型作为对现实世界的抽象，通过一系列科学标准的建模过程，可以有效地对数据特征进行抽象，获取关键信息和指标。目前，数据模型广泛应用于金融、电子商务、电信等行业，其中在信用卡行业的应用颇具代表性。对于商业应用而言，最复杂的模型不一定是最好的，特别是由于企业对于风险的敏感，现实中多采用经过时间考验的相对成熟的建模方法，建模的过程更加强调业务和模型有效融合，因此脱离对业务的理解开发的模型往往会有失偏颇。

数据模型开发的哲学基础是历史的可重复性，即认为事件存在内在的相对稳定的规律。在某种程度上，这种规律主要源于人们的思维方式和行为习惯在某一时期内的稳定性，而人们的社会和经济活动则是这些思维活动的衍生。历史的可重复性使得商业模型的开发成为了可能，并保证了其在未来一段时间内的有效性，然而社会的快速发展和一些突发事件的发生，往往会强烈影响这个基础，因此数据模型并不是万能的，在开发之前需要对应用场景进行深入分析，对于某些重复性欠缺的领域应该寻求其他的解决方案。另外，模型开发实施一段时间，可能由于环境的改变，模型的性能会逐渐发生改变，此时便需要重新评估校准，甚至开发新的模型。

不同于工程领域的仿真模型以精确性作为目的，商业领域所用到的数学模型更多地体现对客户在统计层面的区分情况，即排序性。强调排序性并不是认为精确性不重要，主要是因为在社会学领域影响某一事件的因素往往是非常复杂的，要想精确地刻画一件事往往是徒劳的，较之某一目标的绝对值分布，相对分布往往表现得更为稳定。另外，从业务执行层面来看，模型使用的最终目的通常是实现对客户群的细分，对不同的细分群使用不同的营销策略和管理方案。因此，权衡来看一个排序性很好的模型往往更加现实并能满足业务的需要。

11.1.3 模型开发平台建设

对一个企业而言，基于数据的智能化决策应用是一个系统化的工程，绝不仅仅是开发几个模型的问题，它同时还包括前端的数据准备阶段、后端的策略开发和自动分发部署阶段。数据分析系统建设应包含 3 部分：数据分析集市建设、模型开发和策略部署。其中模型开发部分又细分为开发数据准备、评分卡模型开发和策略设计 3 部分。只有这些部分有机地组合，才能实现从数据到决策的自主和高效。数据决策平台如图 11-2 所示。

（1）数据分析集市建设

数据分析集市的建设主要是以为模型开发和策略实施提供数据支持为目的，涉及数据获取、数据清洗整理和为层级数据集市建设等，通常会将其分为以下几层进行管理。

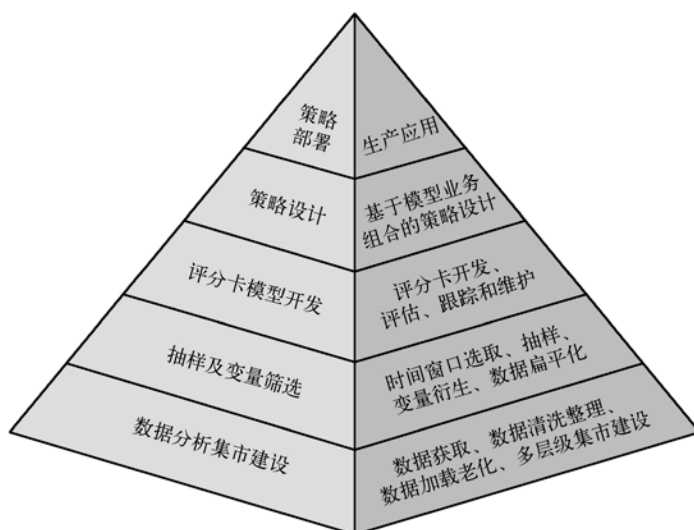


图 11-2　数据决策平台示意图

1）源数据层：存储管理各业务系统的源数据，保证数据的出错回滚和可追溯性。

2）明细层：存储明细数据，对各数据源数据进行抽取、转换、清洗和加载，实现分库存储管理。

3）主题汇总层：以数据主题为主线，建设各主题数据集市，明确各集市的主键、粒度和周期，如产品主题、账户主题和客户主题等。

4）周期衍生层：根据业务常用周期，如分析层的账单周期、报表层的自然月周期，对常用变量进行衍生，生成不同层级（如产品、账户、客户等层面）的扁平化数据集。

5）业务接口层：为各业务策略实施提供接口数据。

分析型数据集市建设的要求：结构清晰、高效稳定、易于维护和便于扩展。

分析型数据集市建设涉及的 SAS 技术难点有以下几个方面：

1）大数据的存储和访问效率。

2）流水表和状态表存储结构设计与读取效率。

3）任务调度方案优化。

4）灵活高效的错误回滚机制。

5）大数据量表的拼接和快速检索技术。

6）Excel 和 TXT 等外部数据的批量导入等问题。

（2）评分卡模型开发

目前，客户关系管理模型中以评分卡模型最为常用。它根据客户各特征变量的取值，对客户赋以一定的分值，然后将各模型变量分值累加作为客户的整体评分，通过该分值来反映客户是好客户和坏客户的概率。

模型开发主要可以分为以下三大部分：

1）抽样及变量筛选。这是模型开发的样本数据准备阶段。首先确定观察窗口（自变量取值）和表现窗口（因变量取值），进行好坏定义；然后对观察窗口数据进行变量准备和变量衍生，并根据观察窗口数据进行好坏标识，实现数据的扁平化工作；接下来结合业务分析设置

变量条件并进行变量粗分组和 WOE 值计算；最后将样本拆分成训练集和测试集，并进行探索性数据分析。数据准备阶段的工作量占据了整个模型开发工作量的一半以上，数据变量的定义是否和业务吻合，以及变量衍生是否充分都直接影响后续模型开发的成败。

2）评分卡模型开发，通常多采用 SAS logistic 回归，用 stepwise 方法进行变量逐步筛选，并根据结果返回去对粗分组等进行调整。模型生成之后，需要进行分数转换形成评分卡，用测试集对模型效果进行评判，最后是模型报表和跟踪监控。

3）策略设计和优化。根据业务需要设定截至分数线，并结合其他变量和模型结果设计冠军和挑战者策略，开发跟踪报表监控模型应用效果。

（3）策略实施

策略实施主要的工作是将结果及时地分发到业务段。根据业务的使用情况，常用的更新周期有每月更新策略（如收益类模型），T+1 更新策略（如客户营销模型）和有交易等触发条件更新的策略（如交易监控模型）。策略实施最重要的是时效性和自动化流程管理。

对于基于评分卡模型的策略，通常以策略树的形式出现，即根据不同的策略变量的取值区间，划分不同的策略分支，最终根据这样一系列变量将客户分发到一定的业务场景。策略树的部署在 SAS 中通常是使用 FORMAT 过程或 HASH 对象构造条件映射，然后将客户变量取值映射到一定的区间。另外，也可以使用 IF 条件语句和区间压缩的方法进行部署。

11.2 数据挖掘建模理论

11.2.1 数据挖掘建模分类

数据挖掘模型开发的思想基础是经验的可重现，即通过对历史上的样本数据集进行模型训练获得模型函数，然后对新样本进行预测，通常都是基于历史学习的判别分类方法。

常见的分类模型有判别分析法、回归分析法、决策树方法、数学规划方法、神经网络方法、最近邻方法和组合预测方法等。其中判别分析法和回归分析法最终主要生成一个评分卡，依据分值对客户进行排序分类，其他一些模型虽然并不生成一个评分卡，但共同之处都是对客户进行细分。在商业应用中常用到的是基于贝叶斯定律的 Logistic 回归方法。

1．判别分析法（Discriminant Analysis）

判别分析是利用已知类别的样本培训模型，为未知样本判类的一种多元统计分析方法。其特点是根据已掌握的历史上每个类别的若干样本的数据信息，总结出客观事物分类的规律，建立判别公式和判别准则。当遇到新的样本点时，只要根据总结出来的判别公式和判别准则，就能判别出样本点所属的类别。

判别分析的步骤如下：

1）根据研究目的确定研究对象（样本）及所用指标。

2）收集数据，得到训练样本。

3）用判别分析法求得判别函数。

4）评估判别函数是否具有使用价值。

5）根据判别函数建立判别准则，对未知样本判别归类。

（1）Bayes 判别法（Bayes Discriminant）

Bayes 判别是一种基于 Bayes 公式

$$P(B|A)=P(A|B)P(B)/P(A)$$

的概率型判别分析。它的基本思想是假设已知对象的先验概率㊀和先验条件概率，而后得到后验概率㊁，由后验概率作出判别。它要求特征变量服从多元正态分布㊂，且两类子总体的协方差矩阵相等。在实际消费信用数据中，这些条件往往不易满足，这是判别分析引起质疑和批评的主要原因。

Bayes 判别过程如下：

考察客户历史样本数据集，存在两个子总体“好”客户（G）和“坏”客户（B），数据集记录了样本的各特征变量的特征值及所属的子总体。

S —— 客户历史样本数据集。

G —— 好客户字总体。

B —— 坏客户子总体。

根据对S数据集的分析，可以得到好坏客户的比例，以及好客户和坏客户的特征变量分布，从而得到以下先验概率和先验条件概率㊃：

P(G) —— 好客户出现的先验概率。

P(B) —— 坏客户出现的先验概率。

P(x|G) —— 特征值 x 的好客户的先验条件概率。

P(x|B) —— 特征值 x 的坏客户的先验条件概率。

新样本数据集是需要预测的样本集，客户是“好”客户或“坏”客户未知，是需要进行预测的因变量，作为自变量的特征变量的特征值分布已知。

S' —— 客户历史样本数据集。

$X=(X_1, X_2, \cdots, X_n)$ —— 特征变量，大写X代表变量。

$x=(x_1, x_2, \cdots, x_n)$ —— 特征值，小写x代表变量取值。

根据历史样本数据集对新数据集进行分类，可以采用 Bayes 判别分析的方法，即通过历史数据集归纳到的先验概率 P(G)、P(B)和先验条件概率 P(x|G)、P(x|G)求后验概率的问题。

从 Bayes 公式 P(B|A)=P(A|B)P(B)/P(A)出发，

$P(G/x)=P(x|G)P(G)$

$=P(G)\cdot P(x_1|G)\cdot P(x_2|(G,x_1))\cdot\cdots\cdot P(x_n|(G,x_1,\cdots,x_{n-1}))$

㊀ 先验概率（Prior Probability）是根据以往经验和分析得到的概率。利用过去的历史资料计算得到的先验概率，称为客观先验概率；当历史资料无从取得或资料不完全时，凭人们的主观经验来判断而得到的先验概率称为主观先验概率。例如，某事件 y 的发生受 x_1，…，x_n 因素影响（$y=f(x_1，\cdots，x_n)$），根据历史样本可以得到在 y 事件发生的概率（先验概率），以及 y 各种取值情况的 x_1，…，x_n 出现的概率（先验条件概率），即先验概率。而对新样本，当某个因素 x_1 出现时，此时判断 y 发生的概率为后验概率。

㊁ 后验概率（Posterior Probability）是指在得到某些信息后重新修正的概率，也就是考虑了一个事实之后的条件概率。后验概率可以通过贝叶斯公式，用先验概率和似然函数计算出来。

㊂ 正态分布（Normal Distribution）又称为常态分布或高斯分布（Gaussian Distribution），若随机变量 X 服从一个数学期望为μ、标准方差为σ^2 的正态分布，则概率密度函数为 $f(x)=\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$。正态曲线呈钟形，两头低，中间高，对期望值μ左右对称，标准差σ决定了分布的幅度，曲线与横轴间的面积总等于 1。通常所说的标准正态分布是$\mu=0$、$\sigma=1$ 的正态分布。

㊃ 先验条件概率就是事件 A 在另外一个事件 B 已经发生条件下的发生概率，条件概率表示为 P(A|B)。 $P(A|B)=P(AB)/P(B)$

先验概率P(G)已知，先验条件概率可以通过客户历史样本数据集的分布计算得到，但可以想象随着特征变量$\{X_1, X_2, \cdots, X_n\}$的个数n的增加，每个特征变量由会对应多个特征值，这样计算工作将会非常繁琐。

（2）Fisher 判别法（Fisher Discriminant）

Fisher判别法是 1936 年提出来的，该方法的主要思想是多维数值到一维数值的映射降维过程，即将多维数据投影到某个方向上。投影的原则是将总体之间尽可能地分开，然后再选择合适的判别规则，将新样本进行分类判别。具体来说，就是针对P维空间的某点$X(X=x_1,x_2,\cdots,x_p)$，寻找一个能使它降为一维数值的线性函数$Y(x)=\sum C_i \times X_i$，然后应用这个线性函数把P维空间中的已知类别总体及未知类别归属的样本都变换为一维数据，再根据其间的亲疏程度对未知归属的样本点判定其归属。这个线性函数应该能够在把P维空间中的所有点转化为一维数值之后，既能最大限度地缩小同类中各个样本点之间的差异，又能最大限度地扩大不同类别中各个样本点之间的差异，这样才可能获得较高的判别效率。在这里借用了一元方差分析的思想，即依据组间均方差与组内均方差之比最大的原则来进行判别。Fisher判别分析也可以被看成一种线性回归分析。

（3）距离判别法（Distance Discriminant）

距离判别的思想是由训练样本得出每个分类的重心坐标，然后对新样品求出它们与各类别重心的距离，从而将其归入离得最近的分类。最常用的距离是马氏距离[㊀]。

距离判别的特点是直观、简单，适合于对自变量均为连续变量的情况进行分类，且它对变量的分布类型无严格要求，特别是并不严格要求总体协方差矩阵[㊁]相等。

2．回归分析法（Regression Analysis）

回归分析是研究一个随机变量Y对另一个(X)或一组$(X_1, X_2, \cdots, X_k)$变量的相依关系的统计分析方法。回归的意义是，认为变量之间存在着某种天然的依存关系，这种关系可以用一个线性或非线性的函数表示回归的过程就是对这个函数的逐步逼近和发现的过程。

回归分析的步骤如下：

1）从一组数据出发确定某些变量之间的定量关系式，即建立数学模型并估计其中的未知参数。估计参数的常用方法是最小二乘法。

2）对这些关系式的可信程度进行检验。

3）在许多自变量共同影响着一个因变量的关系中，判断哪个（或哪些）自变量的影响是显著的，哪些自变量的影响是不显著的，将影响显著的自变量选入模型中，剔除影响不显著的变量，通常用逐步回归、向前回归和向后回归等方法。

4）利用所求的关系式对某一生产过程进行预测或控制。

（1）线性回归方法（Linear Regression）

在回归分析中，若事先假定数据之间满足线性关系，可以用线性函数

$$f(X_1,\cdots,X_n)=A_0+A_1X_1+A_2X_2+\cdots+A_nX_n$$

㊀ 马氏距离是由印度统计学家马哈拉诺比斯（P. C. Mahalanobis）提出的，表示数据的协方差距离。它是一种有效的计算两个未知样本集的相似度的方法。与欧氏距离不同的是它考虑到各种特性之间的联系（例如，一条关于身高的信息会带来一条关于体重的信息，因为两者是有关联的），并且是尺度无关的（scale-invariant），即独立于测量尺度。

㊁ 协方差矩阵的第 I、j 项为 $Cov(X_i,X_j)=E[(X_i-\mu_i)(X_j-\mu_j)]$。

表示，则该回归模型称为线性回归。

普通的线性回归曾被用于解决信用评分中的分类问题，其产生的是一个线性评分卡。但是线性回归方法用于信用评分时存在明显缺陷，即回归方程两边变量的取值范围可能不一致：右边的取值可以从负无穷到正无穷，但方程的左边是概率变量 p，其取值范围只能在（0，1）范围内。如果等式左边变换成 p 的一个函数，它可以取任意值，则模型更有意义。于是，对线性回归进行改进而形成的 Logistic 回归方法就成为信用评分模型中使用最广泛的方法之一。

（2）Logistic 回归（Logistic Regression）

Logistic 评分模型是在商业领域应用最为广泛的预测模型之一，从模型的解释性和稳定性两方面综合衡量，Logistic 模型较神经网络和决策树模型更加适度和均衡。

在 Logistic 回归中，将因变量概率发生比的对数表示成特征变量的一个线性组合，即

$$\log\left(\frac{p_i}{1-p_i}\right)=A_0+A_1X_1+A_2X_2+\cdots+A_nX_n$$

Log 函数的作用主要有两个方面，首先将因变量和自变量之间的非线性关系变换成线性关系；其次将概率 p 的区间[0,1]映射到[−∞,+∞]，同等式右边保持一致。因变量可以是二分类的，也可以是多分类的，但是二分类的更为常用，信用评分中即‘好’和‘坏’。对于因变量是连续的情况通常通过一定的方法将其转换为二分变量。

Logistic 回归模型克服了线性回归模型的缺陷，其回归方程两边的值均可取任意值。就理论背景而言，人们会认为在信用评分中 Logistic 回归比线性回归更合适，而 Logistic 回归模型也是现实中应用最广的评分模型。

3．决策树/分类树方法（Decision or Classification Trees）

分类树方法最后不生成一个评分卡，而是将消费者分成不同的组，在组内各样本的违约概率尽量相等，而违约概率在组之间的差异则尽可能大。其特点是能更有效地处理特征变量之间存在相互作用的情形，而且即使有些特征变量存在一定的数据缺失，该方法也能适用。另外，分类树方法可以通过人为设置节点从而实现业务经验的嵌入，建模速度快，适合精确性要求较低的领域。分类树方法也有一些缺陷，如某些低端节点所包含的样本可能太少，从而使得在这些节点中所作的统计推断不可靠。

4．数学规划方法（Mathematical Programming）

数学规划方法通过研究对客户信用有影响的各个因素并确定它们的权重，把客户分为好、坏两类，从而建立一个线性规划方程，目的即使方程误差最小。它也产生一个线性评分卡。绝大部分文献认为线性规划方法与统计学方法效果相当。

5．神经网络方法（Neural Network）

神经网络是一种模仿人脑信息加工过程的智能化信息处理技术，具有大规模并行处理、自学习、自组织、自适应等能力，能独立处理复杂的非线性问题，不限于严格的前提假设条件。该方法常应用于风险管理中，能够较好地忠实于客观实际，对噪声和缺失数据有较强的忍耐能力。

Chen 与 Titterington(1994)认为，神经网络方法实际上可以看做一种非线性回归。该方法可能存在过度拟合的问题。Davis(1992)也比较过神经网络与其他方法，认为神经网络能很好地处理数据结构不太清楚的情况，但其训练样本时间较长。此外，其可解释性较差也受到质疑。

神经网络与判别分析最大的差异在于神经网络具有学习能力，因此对于无法以线性模型来区别群组的问题，神经网络最能发挥特长与优势。判别分析是一种“白盒”技术，具有较强的透明度，模型中变量的系数都具有一定的经济学含义，代表了指标的重要程度，可以比较各变量对模型贡献的大小。而神经网络是一种“黑箱”技术，它可以根据新的样本不断地调整模型，适用于样本分布不断变化的情况。根据以往的研究，神经网络方法能很好地处理那些数据结构不太清楚的情况，且短期预测准确性稍好于判别分析方法，但其工作随机性较强，为了得到一个较好的神经网络结构，需要人为地去调试，而且其样本训练时间较长，非常耗费人力和时间，因此在计算效率、可解释性、适应性、稳定性、操作简单性方面逊于判别分析方法。此外，神经网络在分类不当的情况下错判比例较高，可能造成模型的解释性不强。

神经网络的优点是预测精度较高。其缺点如下：①稳健性不够好，当用保留样本进行预测时，精度下降较多；②模型的解释性不强，建模过程基本上是一个“黑箱”；③它主要用于分类，即将申请人分成“好客户”和“坏客户”，而不能像判别分析那样产生线性评分卡。判别分析方法的主要优点是稳健性较好，模型的可解释性较强，可以产生一个线性评分卡；其缺点是预测精度比神经网络差。

6．最近邻方法（Nearest Neighbour Classification）

最近邻方法是一种非参数方法，它的思想是在客户的特征向量空间内定义一种测度（距离）用于测量两个客户之间的距离。当对一新客户信用评估时，只要考察与他最近邻的 k 个人中“好客户”及“坏客户”的比例，根据此比例确定该客户的信用类型即可。

7．组合预测方法（Combination Forecasting）

组合预测方法就是设法将不同的预测模型组合起来，综合利用各种预测方法所提供的信息，以适当的加权平均形式得出组合预测模型，以提高预测性能。

例如，利用申请信息样本建立信用评分模型 S1，再从信用局取得这些客户的信用局评分 S2，然后利用两种评分构造一个组合评分 SC=A0+A1S1+A2S2。通过选择适当的组合系数，组合模型评分就可能优于单个模型评分。

11.2.2 评分卡模型分类

（1）评分卡模型应用历史

1936 年 Fisher 提出了线性判别分析法，即基于 Fisher 线性判别函数产生一个线性评分卡，1941 年杜兰特（Durand）首先将这一思想应用于信贷领域，区分“好”的贷款和“坏”的贷款。

20 世纪 60 年代后期，随着信用卡的出现和发展，银行和其他信用卡发卡机构开始认识到信用评分的作用和重要性。

20 世纪 80 年代随着信用评分方法在信用卡领域应用的成功，银行开始将信用评分应用于个贷等其他金融产品。

20 世纪 90 年代直销公司开始使用评分方法改进广告销售中的响应率。国内银行从 2000 年开始逐渐使用信用评分。

（2）评分卡模型分类

数据挖掘建模在商业应用中以评分卡应用最为典型，根据评分卡的目标对象和应用场景

可以化分为不同的类别，但开发方法基本一致。

根据评分对象评分卡模型可以分为以下两种。

1）产品水平：以不同客户的不同产品作为主键形成开发数据，如银行中对持有某类理财产品客户的收益的预测，在这种模型中主要使用的是该产品账户的行为信息。

2）客户水平：对于一个客户拥有多个产品账户，如银行中某个客户既有贷款账户，又有借记卡和理财产品，这种情况下若银行需要开发风险类评分卡时，往往需要综合考虑该客户的多个产品的信息，那些同时持有借记卡的贷款客户的风险通常较仅有贷款的客户的风险水平偏低。

根据应用目的评分模型可以分为以下 3 种。

1）风险评分：主要对客户贷款或信用卡等在未来的一段时间内的逾期可能性进行预测，通常评分越高，逾期的可能性越高。

2）收益评分：主要根据客户使用企业产品的行为变量，对客户未来一段时间内可能产生的收益高低进行预测，通常评分越高，收益越高。在金融类收益评分卡中，收益指标的定义通常指收入扣除资金成本，而不是扣除其他成本后的利润，这样做的目的是为了避免造成模型开发的复杂化。另外，收益评分卡结果通常会和风险评分卡结果交叉使用，对不同客户群使用不同的策略。如图 11-3 所示的风险收益交叉策略中，其中低风险高收益的客户为企业的核心客户群，应该全力维护；对高风险低收益的客户应该采取一定的限制措施；低风险低收益的客户通常以新客户较多，应以培养客户的产品使用情况为主要目的，采取活动促动；高风险高收益的客户最需要关注的是其风险变化情况，目标是将此类客户的风险控制在可接受的范围。

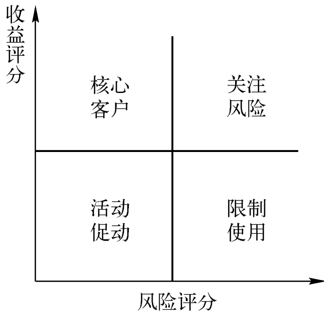


图 11-3　风险收益交叉策略

3）流失评分：主要对客户未来一段时间的流失情况进行预测。通常评分较高的客户的流失可能性较高，流失评分结果同样可以和收益评分结果交叉应用。

根据客户生命周期管理的需要，评分模型可以分为以下 3 种。

1）申请类评分：这类评分模型用在客户生命周期管理的客户获取阶段，被用来定位需要拓展的客群。由于客户获取阶段关于客户的行为变量是缺少的，因此开发申请类评分模型用到的变量主要是客户的属性变量，数据主要来自企业内部的历史数据和人行征信等外部数据，基于模型开发样本同预测样本分布的一致性要求，申请类评分模型开发时需对申请拒绝的客户表现变量进行拒绝推断。申请评分卡常用的有申请收益评分卡和申请风险评分卡。

2）响应类评分：这类评分模型用在客户生命周期管理的客户培养阶段，主要用来预测客

户对特定促动活动的响应。响应类模型可以分为单产品响应模型和多产品比较响应模型，开发模型用到的变量信息有客户的属性变量、客户的行为变量和产品信息等。

3）行为类评分：这类评分模型用在客户生命周期管理的客户管理阶段，主要用来预测客户的风险、收益和流失等情况。由于行为类评分卡主要是根据客户的行为习惯来对未来预测的，因此开发此类评分卡所使用到的变量应以行为类变量为主，尽量避免使用属性类变量。在具体的策略应用中可以结合申请类评分卡和行为类评分来对一个客户进行完整的评价。从某种程度可以说申请类评分反映的是一个客户的潜在能力，而行为类评分反映的是一个客户的真实使用情况。例如，一个客户的申请收益评分很高，而行为收益相对较低，则很有可能该客户持有其他公司的同类产品，基于此可以制定对应的管理策略。

11.3 数据挖掘建模流程

数据挖掘建模流程可以分为以下 6 个阶段：需求分析、数据准备、模型开发、模型验证、策略设计和模型部署，如图 11-4 所示。

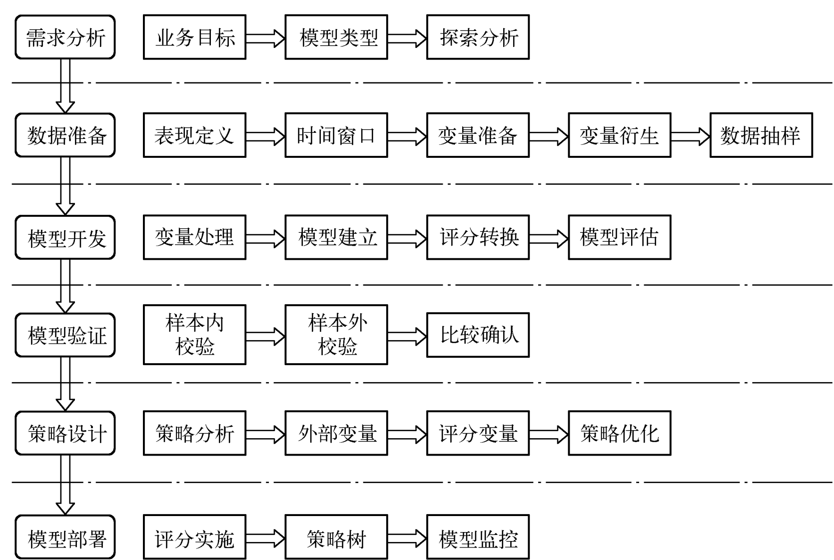


图 11-4　数据挖掘建模流程

11.3.1 需求分析

目标：明确业务需求，确定模型类型。

需求分析是建模的第一步，是整个建模的纲领性部分。它主要包括 3 个部分：业务目标、模型类型和探索分析。通常需求分析部分由业务部门和分析建模团队共同合作完成，甚至需要模型开发人员深入业务第一线进行实地调研，这些工作将会有利于模型人员深入了解影响商业目标的因素，培养业务直觉，对后续变量的选择、衍生和调整起着重要的作用。

1．业务目标

业务目标部分主要是提出问题和确定问题的过程，解决的是模型要预测什么的问题。

企业中的业务部门（市场部和风控部等）在日常作业中会伴随一些问题，在没有模型依据的时候通常的解决方案是根据业务经验进行决策，这样就会导致政策的随意性和不确定性。常见的业务问题有以下几个方面：

1）如何发现并精确定位营销的新目标客户群？

2）针对不同客户群，应该向客户推荐哪些产品？

3）对已持有某类产品的客户，如何提高交叉销售效果？

4）如何对存量客户开展促动活动才能达到最好的效果？

5）如何发现高风险的客户？

6）如何发现有价值的客户？

7）如何提前发现有流失倾向的客户？

当业务人员提出业务问题后，分析建模人员应对这些问题进行建模可行性分析，这些分析包括基于企业数据仓库数据的了解，确定是否存在相应的历史样本。例如，如果要对客户进行某类促动活动时，就必须在历史上曾经做过此类或类似类型的活动。如果缺乏数据，则必须首先进行小范围的测试以便积累数据。

2．模型类型

当业务人员提出业务问题，并确认可行性之后，分析建模人员需要将相应的业务问题进行抽象化，从而确定需要开发的模型类型。这个过程通常需要反复多次，经过不断的修正，最终在业务人员和分析建模人员之间达成一致。对应上述业务问题的模型类型如下：

1）申请类评分模型。

2）单产品响应类模型。

3）多产品响应类模型。

4）活动响应类模型。

5）风险评分卡。

6）收益评分卡。

7）流失评分卡。

3．探索分析

探索分析的主要目的是了解相关业务的历史情况和数据分布情况。通常由分析建模人员根据时间维度对相关业务主题的数据进行统计，由业务人员整理相关的大事记，通过数据和业务记录的比对，深入了解数据变化同业务之间的联系，从而有效剔除人为因素对数据的影响，尽量选择相对“干净”的时间段的数据。

另外，通过对数据探索分析可以了解到历史上该业务的相关效果，使分析建模人员对模型效果形成初步的预期。

11.3.2　数据准备

目标：扁平化开发数据集。

数据准备阶段是建模过程中最为重要的阶段，数据变量的衍生质量将会直接影响到建模的质量，数据准备阶段的时间通常占据整个模型开发过程的一半以上。这里假设企业已经建立了较为完善的分析数据集市，数据准备阶段可以进一步细分为 5 部分：表现定义、时间窗口、变量准备、变量衍生和数据抽样。

1．表现定义

表现定义主要是对目标变量进行好坏定义，在某些业务场景下好坏定义是比较显然的。例如，在响应类的模型中，只要对某种营销活动进行了回应和参与，即可定义为好客户，而那些对营销活动没有反应的客户就可以定义为坏客户；在申请类模型中，通过审批的客户即为好客户，未通过审批的客户即为坏客户。然而在另外一些业务场景中好坏定义就变得相对比较模糊。例如，在收益类评分模型中，由于收入值为连续变量，无法明确定义好坏，通常的做法是将连续变量进行离散化，将最高收益定义为好，将最低收益定义为坏，而将客户视为部分好部分坏进行处理。

表现定义涉及的另外一个重要的问题是表现窗口长度的选择，即根据客户在多长时间内的表现来定义客户是好是坏。表现期的选择通常不宜过长，以免目标变量同观察变量之间的时间跨度太大，同时表现期的选择必须使客户的行为得以充分表现。例如，在产品营销类的模型中，通常选择 3 个月的表现期；收益类评分模型通常为未来 6 个月或 12 个月的收入情况；风险及流失类的评分卡要做逾期/销户累计分布分析以确定选取多长的观察期。一般情况下，当累计占比达到 80%以上，且增长趋缓所对应的时长便可以被选作表现期长度。

当表现定义和表现窗口确定以后，需对表现窗口中的客户进行分析，剔除一些影响建模效果的客户。常见的表现剔除有表现期销户的客户，表现期逾期等状态异常客户，表现期数据缺失无表现的客户。

2．时间窗口

时间窗口的划分用来设定观察变量和表现变量的取值范围，其主要包括 3 个要素：观察点、观察窗口和表现窗口，如图 11-5 所示。

观察点：观察点是用来对客户进行评分的时点，通常为月末时点、账单日或活动发起日期。

表现窗口：在时间轴上从观察点向后推得到表现窗口，用来提取目标变量和进行表现排除。

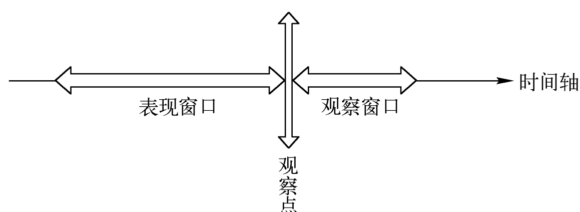


图 11-5　时间窗口示意图

观察窗口：从观察点向前推一段时间得到观察窗口，用来提取自变量信息和进行观察窗口排除，观察窗口的长度通常为一年，以消除季节性因素的影响。

3．变量准备

时间窗口确定完毕，根据响应时间段分别从数据集市中选取相关的主题数据，然后对数据进行探索分析和处理，生成基础变量各个月度的数据集。

在商业分析中常用的主题数据集市如下

1）客户属性数据：存储客户的自然属性和个人收入、兴趣、偏好、产品类型等的数据集

市，主键通常是客户的证件号。

2）产品信息数据：存储客户各个产品的基本信息和状态变更情况。主键通常是客户证件号和产品号的组合，或企业内部定义的账户。

3）客户交易数据：客户的交易信息，包括交易金额、交易日期、交易币种、交易地点和商户信息等。交易表是客户行为变量的最重要来源。

4）客户账单及风险数据：反映客户的账务信息和逾期情况。如客户账单余额、额度使用率、逾期金额和逾期期数等信息。

5）营销活动数据：记录客户对营销活动的响应情况。

根据模型开发的层级，如产品级，将各个数据集市可能用到的基本变量汇总，然后将所有的汇总后的数据按产品和时间主键进行横向合并，生成由原始变量构成的数据集，即原始变量的扁平化。

对原始变量的扁平化数据集进行各变量的分析，如频数、分布、缺失情况和异常情况等。其中缺失和异常需要区别对待，异常一般表示数据存在错误，而缺失本身也代表一定的意义。例如，申请表中的婚姻状况缺失代表对该信息敏感。在数据充分的条件下通常将存在较大比例缺失和异常的变量直接删除，而不是进行填补。

4．变量衍生

变量衍生主要是在原始变量的扁平表基础上衍生出可能进入模型的一些变量，某种程度上变量的衍生也可以被认为是一种去除变量非线性的方法。常见的变量衍生方法如下。

1）最近值：距观察点最近 n 个周期的值，如最近一个月的交易金额，最近两个月的交易金额等。

2）趋势值：反映观察区间某指标变化方向的变量，如交易金额连续递增的最大月数，交易金额连续递减的最大月数等。

3）均值类：反映客户平均水平的变量，如近半年平均交易金额等。

4）偏差类：反映客户稳定水平的变量，如近半年月交易金额的方差等。

5）峰值和谷值：反映客户的极值情况，如最大交易金额、最低交易金额等。

变量衍生首先将原始数据集在时间维度进行转置，使得每个样本对应一条记录，然后在此基础上通过构造数组的方式，使用不同的函数得到衍生变量，最终形成模型开发所需要的扁平化的开发数据集。

5．数据抽样

进行抽样前，首先根据时间窗口分析确定的排外条件，对上一步扁平化的开发数据集中的记录进行样本排除，得到开发样本集，然后从其中按照随机的方式进行抽样。通常上述样本集在抽样过程中会被按照 7:3 的比例被分成两个部分，70%的样本用来进行模型的训练，剩余 30%的样本被用来做后续样本内的模型校验。

抽样的目标通常使得样本中好坏客户比例相当，如果总样本集中坏客户过少，通常的做法是将全部坏客户均抽取，好客户进行随机抽取，然后对记录数添加权重因子，以保证同总样本的分布一致。例如，银行中的风险类评分卡开发中，形成坏账的客户占总体客群的比例通常非常小，抽样是将其全部选取，而从好客户中选择相应量的好样本即可。

最后，值得注意的是，为了保证后续各变量特征值分组的客户数具有统计意义，一般样本的数量要求大于等于 5 000 个。

11.3.3　模型开发

目标：得到评分卡。

模型开发主要分为变量处理、模型建立、评分转换和模型评估 4 个步骤。

1．变量处理

基于抽样后得到训练样本集数据，由于变量数量通常很多，不推荐直接采用逐步回归的方法进行变量筛选。另外，由于各变量的量纲和取值区间存在很大差别，通常会先对变量的取值进行分箱并计算证据权重 WOE（Weight of Evidence）值，从而降低变量属性的个数，并且平滑变量的变化趋势。接下来，在此基础上计算信息价值 IV（Information Value），一般选取 IV 值大于 0.02 的那些变量进入模型。如果当一个变量的 IV 值大于 0.5 时，该变量属于过预测变量，通常被选作分群变量，将样本拆分成多个群体，针对不同的群体分别开发评分卡。分群的依据通常也会根据业务上的需要进行设定，常见以区域变量作为分群的标准。

证据权重 WOE 是用来衡量变量某个属性的风险的指标，WOE 的计算公式为

$$\mathrm{WOE_i} = \ln\left(\left(\frac{g_i}{g}\right)/\left(\frac{b_i}{b}\right)\right)$$

式中　$\mathrm{WOE_i}$ —— 某变量第i各属性对应的WOE值；

g_i —— 某变量第i各属性对应的好客户数；

b_i —— 某变量第i各属性对应的坏客户数；

g —— 样本中总的好客户数；

b —— 样本中总的坏客户数。

WOE 的值越高，代表着该分组中客户是坏客户的风险越低。

信息价值 IV 是用来衡量某个变量对好坏客户区分能力的一个指标。IV 的计算公式为

$$\mathrm{IV} = \sum_i\left(\frac{g_i}{g}-\frac{b_i}{b}\right)\times\log\left(\left(\frac{g_i}{g}\right)/\left(\frac{b_i}{b}\right)\right) = \sum_i\left(\frac{g_i}{g}-\frac{b_i}{b}\right)\times \mathrm{WOE_i}$$

IV 的值越大表示好坏客户在该变量上的分布差异越大，也即该变量的区分能力越好。

2．模型建立

评分卡模型用到的方法很多，其中 Logistic 回归是应用最为广泛的方法。对于 Logistic 回归模型，目标变量是二元值的概率函数，即前面定义的好坏客户的概率。Logistic 回归分析的结果并不是对一个客户给出明确的好或坏的概率，而是该客户是好客户或坏客户的概率，是一个连续的值。具体公式如下所示：

$$\log\left(\frac{P_G}{P_B}\right) = \log(\mathrm{odds})$$

即目标函数可以看做是对 WOE 值的线性回归函数。

将转换后的变量的 WOE 值作为自变量输入建模，SAS 编程过程使用 PROC LOGISTIC 过程步来完成建模。

```
proc   logistic   data=INPUT_TB;
       model   Y(event='1')=X_LIST
                   /selection=stepwise
                    slentry=0.05
```

```
            slstay=0.05
            details
            outroc=roc;
        weight    weight;
    run;
```

【程序解读】

data=：指定用于该过程的分析建模数据集为 INPUT_TB。

model：指定模型中的响应变量和自变量。

Y(event='1')：Y 是响应变量，为二分变量{0,1}分别代表坏客户和好客户，event='1'代表建模的对象是好客户的概率。

X_LIST：自变量列表。

selection：指定变量筛选的方法。

变量筛选的方法有以下 3 种。

1）FORWARD——向前回归法：初始选择一个变量进入模型，后续变量逐个加入模型，通过计算该变量的偏回归平方和，决定是否保留该变量。向前回归法也被称为进入法，变量一旦被选择，就不会再被剔除。

2）BACKWARD——向后回归法：初始选择所有变量进入模型，然后根据偏回归平方和逐个删除不显著的变量。向后回归法也被称为退出法，变量一旦被剔除，就不会再被选用。

3）STEPWISE——逐步回归法：最常用的方法，变量逐个进入模型，进入时进行偏回归平方和检验，当该变量被选中，需对原有的变量重新进行偏回归平方和检验，并剔除不显著的变量。最后直到其他新变量再也无法进入模型，同时老变量再也无法被剔除为止。

slentry：当 selection=stepwise 或 forward 时指定变量进入模型所要求的显著水平。

sltay：当 selection=stepwise 或 backword 时指定变量保留在模型中的显著水平。

details：输出模型选择过程的每一步的详细结果。

outroc=：对二值响应模型输出一个数据集，如数据集名 ROC。其包含生成 ROC 曲线的相关数据。

weight：指定数据集中的每一条观测的权重系数，如变量 weight。weight 可以是小数，若 weight 的值为负数或缺失，那么忽略该条观测记录。

3．评分转换

模型建立以后得到的 log(odds)值是建模样本的好/坏比的对数，分值可以为负值，使得分值的可解读性很差。为了使评分结果更加容易理解，具有实用性，我们希望看到的评分卡均为正整数，因此通常对变量的特征值进行线性比例变换，并加上一个偏移量，如下所示：

$$\begin{aligned}\text{SCORE} &= \text{factor} \times \log(\text{odds}) + \text{offset} \\ &= \text{factor}\left(\sum_i a \times \text{woe}_i + b\right) + \text{offset} \\ &= \sum_i \omega_i \times \text{woe}_i + \beta\end{aligned}$$

其中，比例因子 factor 和偏移量 offset 可以通过过以下常用规则确定：

1）好：坏=20:1 时对应评分 600。

2）评分每增加 20 分，好坏比增加一倍。

因此有：

$$\log(20)\times factor+offset=600$$
$$\log(40)\times factor+offset=620$$

解得：

$$factor=20/\log(2)= 28.8539$$
$$offset=600-\log(20)\times factor=600-20\log(20)/\log(2)= 513.561$$

所以有：

$$\begin{aligned}SCORE&=28.8539\times \log(odds)+513.561\\&=28.8539\times\left(\sum_i a\times woe_i+b\right)+513.561\end{aligned}$$

通过上式可以根据变量各特征属性的 woe 值及回归系数得到评分值。

4．模型评估

模型评估主要是对模型的效果进行评价，通常包含 3 个方面：模型的合理性、模型的预测能力和模型的稳定性。

（1）模型的合理性

模型的合理性主要指评分结果同业务经验要吻合，不存在严重的背离。合理性的验证通常通过变量和评分的描述统计来分析，如最大值、最小值和趋势等。

（2）模型的预测能力

模型的预测能力指评分卡能够对目标变量具有较好的预测，即模型对好坏客户具有一定的区分能力，且同实际情况吻合度较高。常用的衡量评分卡预测能力的指标有分离度（Divergence）、K-S 检验（Kolmogorov-Smirnov）和 GINI 系数等。

分离度（见图 11−6）主要是基于评分的好坏客户的整体分布的差别，计算公式如下：

$$divergence=\frac{(\mu_{good}-\mu_{bad})^2}{\frac{1}{2}\left(\sigma_{good}^2+\sigma_{bad}^2\right)}$$

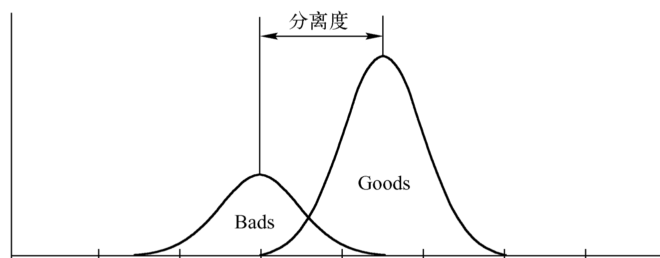


图 11−6　分离度（Divergence）

通常分离度越高，评分模型的排序能力越强。

K-S 值（Kolmogorov-Smirnov）描述的是好坏客户群体累计分布的最大差异，如图 11−7 所示。

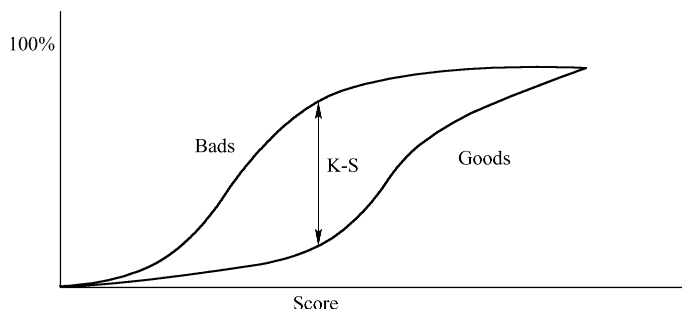


图 11-7　K-S 值（Kolmogorov-Smirnov）

通常 K-S 值越大，评分模型的排序能力越强。

GINI 系数是好客户累计百分比与坏客户数累计百分比在模型与随机时的差值，如图 11-8 所示。

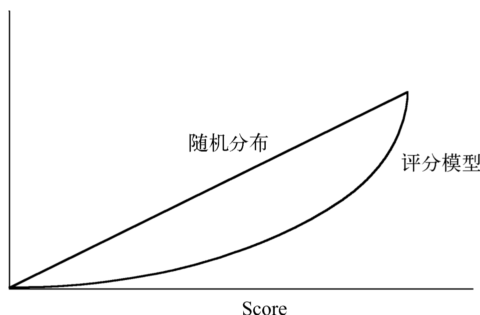


图 11-8　GINI 系数

（3）模型的稳定性

稳定性的检查主要通过后面模型验证部分的样本内校验和样本外校验来完成。

11.3.4　模型验证

目标：确认模型对新样本有效。

模型验证是为了检查从训练集的学习得到的模型是否对新的样本数据有效，主要可以分为 3 部分：样本内校验、样本外校验和模型比较确认。

（1）样本内校验

在模型开发前的数据准备阶段，抽样将总体拆分成 7:3 两部分，其中 70%的样本用来作为训练集进行模型学习，剩下的 30%被用来作为模型校验。由于这两部分来源于一个总体，因此又称为样本内校验。样本内校验回答的是模型对于同一时期的新数据是否有效的问题。样本内校验的步骤如下：

1）对该部分数据中和模型相关的变量的特征值，按照模型中该变量特征值和 WOE 的对应关系进行映射，得到该变量的 WOE 值。

2）根据评分转换公式，结合步骤 1）中得到的 WOE 值计算样本的评分。

3）对新样本的评分结果进行模型评估，通常只选择 K-S 值与训练样本进行比较，检查

K-S 值是否发生较大变化，若变化不大说明模型稳定性较好。

（2）样本外校验

如果说样本内校验是对模型空间有效性的校验，那么样本外校验则是对模型在时间维度上的有效性的检验。可以想象，随着时间的推移带来的环境因素的改变，模型的适用性势必会有所降低，但对于模型的应用通常要求其在一定的时间段内具有相对的稳定性。通常一个模型的使用寿命通常为两年，样本外校验使用的数据通常为训练集中观察点后移 3～6 个月。

样本外校验的过程同样本内校验基本类似，所不同的是需要根据新的观察点确定观察窗口和表现窗口，并生成新的扁平化数据集。

（3）比较确认

在模型开发阶段，通常会出现效果类似的模型变量组合，有时会有人为的对于变量的调整，这样一来会存在多个待选样本，比较确认就是通过上述的分析方法比较各个样本的合理性、预测能力和稳定性，最终做出选择，确定综合效果最优的模型。

11.3.5 策略设计

目标：生成基于模型业务策略。

策略设计是根据评分模型的结果，划定分数区间，并结合其他业务变量对客户进行细分，形成不同的营销和管理策略，最终在业务层面解决业务需求问题。策略设计主要包括策略分析、外部变量、评分变量和策略优化。

（1）策略分析

首先根据业务需求确定业务要实现的目标，并分析业务活动的方案和限制条件，最终确定如何将使用模型结果进行策略设计，以产品推荐策略为例。

业务目标：向不同的客户推荐最有可能得到客户响应的产品，获取最大的响应率。

活动方案：礼品馈赠、服务馈赠、费用优惠等。

限制条件：一定限额的营销成本。

根据上述分析，首先利用产品响应模型得到的客户对不同产品的响应评分，从中挑选响应评分最高的产品作为推荐产品。其次，对相同推荐产品的不同客户根据其评分高低，设置合适的分割点，将客户划分为 3 个区间，针对不同区间分别使用不同强度的营销方案。例如，对响应评分较低的客户直接进行礼品馈赠，响应评分中等的使用费用优惠，而响应评分较高的可以进行服务馈赠或不采取任何营销。最后，根据活动成本限制，核算目标人数，从各分群中按照从高到低的评分进行客户筛选，完成整个策略的分析和制定。

（2）外部变量

业务策略中除了会使用模型相关的变量外，还会根据政策和实际执行限制选取一部分外部变量作为决策依据。最常用的外部决策变量有地域、国籍等变量。例如，银行中的很多银行业务通常以分行为具体执行单位，不同地区的政策存在很大的差别，因此政策设计时通常会首先根据区域变量进行划分。另外，外部变量还包含其他模型评分结果变量。总之，在策略准备阶段需要完成相关外部变量的分析和准备工作。

（3）评分变量

评分变量是指模型的评分结果，被用来对客户进行排序。评分变量使用方法通常有单点切割和多点切割两种方法。

单点切割是根据评分分布设置一个评分截止点，高于该评分的客户通过，而低于该评分的客户则被拒绝。例如，申请评分中通常会根据预先容忍的坏账率，计算出相应的申请评分点，高于该评分的客户会直接通过审批。

多点切割是根据评分分布设置多个分割点，将所有客户划分成两个以上的群体。多点切割的使用多见于响应类评分卡。例如，流失挽留策略中，通常业务部门会设置多套不同力度的挽留方案，而评分应用中会对应将客户分成多个群体，对评分较高的人群使用促动力度相对较弱的方案，而对评分较低的人群使用促动力度较强的方案。

（4）策略优化

策略优化的方法有单因素实验方案和冠军挑战者方案。单因素实验是在每一个实验组中控制其他变量不变，只改变一个变量的取值，以测试该变量对策略结果的影响。单因素实验可以准确地反映每个因素对策略的影响，但不能反映不同因素之间可能存在的相互影响。

冠军挑战者方案是最常用的方法，首先将当前最好的营销策略作为冠军者策略，然后根据模型结果制定相对冠军策略有一定程度偏离的策略。挑战者可以是一个也可以是多个。接下来将应用客群随机分成两组或多个实验组，其中一个小组应用冠军者策略，其他小组应用挑战者策略，根据后续跟踪策略执行结果，对比不同小组的客户表现，如市场响应率等，将响应率最高的小组的策略作为冠军。这样通过不断的实验来测试新的挑战者策略，最终得到最优化的策略方案。通常由于成本和风险的考量，冠军策略的小组比例占整个群体的 80%以上，而挑战组策略在具有统计意义的情况下尽可能的限制人数。冠军挑战者方案从最终效果评估整个策略的优劣，但很难给出每一个变量的策略结果的影响程度。

11.3.6 模型部署

目标：实施和跟踪模型策略。

模型的部署是将评分卡模型和策略树自动部署到企业的生产运营系统中，实现每日或一固定周期策略的自动生成、分发和效果监控工作，主要可以分为 3 部分：评分实施、策略树和模型监控。

1．评分实施

评分实施解决的是从模型变量到客户评分分值转换的任务。首先，基于评分模型生成的评分卡构建各变量区间到分值的映射，在 SAS 中可以通过 PROC FORMAT 过程或者 HASH 对象的方法实现。然后，将映射应用到各个模型变量，并对各变量的评分进行累加得到客户的整体评分。

2．策略树

策略树解决的是从描述策略——树形结构——策略实施的任务，也即如何将策略部署到生产系统中。这个过程主要考虑的是策略的变更需要的维护便利性。

（1）策略树的生成

策略树的生成其实就是描述性策略到策略树形结构的转换。描述性策略通常是由一系列的条件语句组成的。例如，年龄大于等于 18 岁可以允许贷款审批，而年龄小于 18 岁的不能获得贷款。这些自然语言的描述通俗易懂，但缺乏标准化，需将其转化标准的描述方式，以便生成策略参数表，便于部署和维护。通常所用的方法是将其转化为树形结构，如图 11-9 所示。

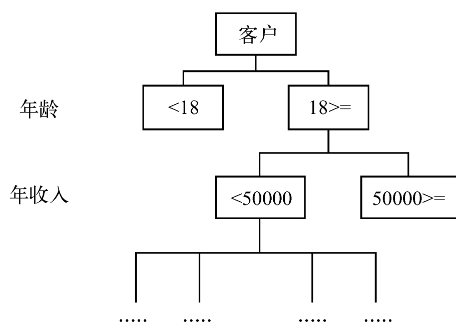


图 11-9　策略树结构示意图

（2）策略树的部署

策略树的部署主要是指将策略树转化成参数表，并将参数表应用到开发系统的工作中。从策略树到参数表通常的做法是将策略树的每一行对应到数据表的一条记录，并且上一条记录的区间覆盖下一条记录对应的子区间。图 11-9 的策略树经过转化如表 11-1 所示。

表 11-1　策略参数表

varnm	value1	value2	value3	value4
年龄	0	18		∞
年收入		0	50000	∞

3．模型监控

模型监控主要包含两部分：评分模型的监控和策略结果的监控。

（1）评分模型的监控

每一个评分卡都存在相应的使用寿命，随着应用环境的变换，模型预测的前提历史的重现性不再存在，因此需要对模型的应用效果进行跟踪、分析、对比和评价，决定模型是否还可以继续使用，是否需要对模型进行调整和重新开放。通常在模型部署上线之后，会开放一系列的监控报表分别对模型进行前端和后端的跟踪。

前端跟踪主要解决模型的适用性问题，即评估应用评分卡模型的人群分布和开放评分卡时的样本人群分布是否发生重大偏移。常用的评价指标有 PSI 和 CSI。

后端跟踪主要解决模型的有效性问题，即评估模型在新样本人群中的预测结果和实际表现结果之间的差异。常用的评价指标有分离度和 K-S 值。

（2）策略结果的监控

策略监控主要对冠军和挑战者策略结果进行跟踪，评价新的冠军策略，进行策略的重新调整。

11.4　评分卡模型开发案例

本节以响应评分卡模型的开发为例具体介绍评分卡开发中的 SAS 相关内容。响应评分卡模型是针对客户对某一种产品的响应概率进行预测的模型。

11.4.1 前段准备

1．需求分析

（1）业务目标

银行新推出一种理财产品，计划向银行目前所有存量客户进行营销，需要对其中响应率较高的客户重点推荐，对其中响应率较低的客户短信通知。

（2）模型类型

单产品响应模型。

（3）探索分析

对历史数据进行分析，选择类似产品的营销的相关数据构建建模数据集。

2．数据准备

（1）表现定义

在历史营销活动中，定义购买产品的客户为好客户、没有购买该产品的客户为坏客户。

（2）时间窗口

观察点：历史活动开始日期，如 2012 年 1 月 1 日。

表现窗口：首先对活动响应事件进行累计客户数分析。客户在营销时点后 6 个月内的累计客户数变化情况，如图 11-10 所示。

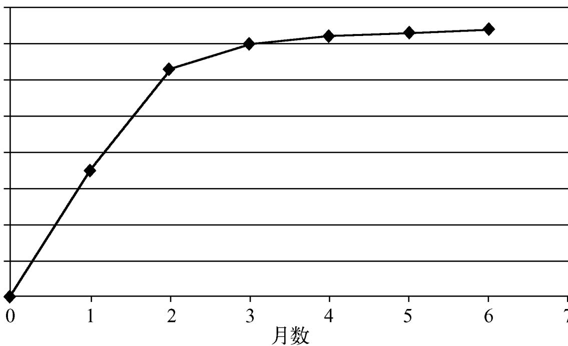


图 11-10　客户累计响应分布图

从图 11-10 可以看到在第三个月时有约 95%的响应客户已经响应。也就是说，3 个月的时间客户的响应的好坏情况已经充分表现，因此可以将表现窗口的长度定义为 3 个月来对客户的好坏进行定义。

表现窗口为 2012 年 1 月 1 日～3 月 31 日。

观察窗口：观察窗口通常选择客户观察点向前一年的时间，这样可以避免节假日等季节性因素的影响。观察窗口为 2011 年 1 月 1 日～12 月 31 日。

建模的时间窗口如图 11-11 所示。

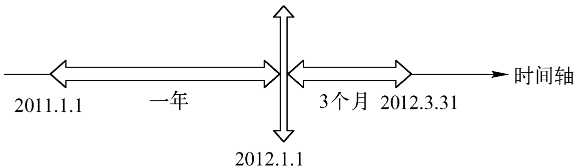


图 11-11　时间窗口

（3）变量准备

建模可能用到的主题集市及数据。

1）客户属性数据表如表 11-2 所示。

表 11-2　客户属性

序　　号	变 量 名 称	变 量 标 签
1	user_id	客户 ID
2	sex	性别
3	age	年龄
4	home_address	户籍地址
5	work_address	工作地址
6	family_address	家庭地址
7	phone	电话
8	marriage	婚姻状况
9	house	住房状况
10	occupation	行业
11	postion	职位
12	income	收入
	…	……

2）账户产品数据表如表 11-3 所示。

表 11-3　账户产品属性

序　　号	变 量 名 称	变 量 标 签
1	acct_id	账号
2	product_id	产品 ID
3	type	账户类型
4	Level	账户级别
5	mtgloan_bal	按揭余额
6	deposit_bal	存款余额
7	deposit_yravg	存款年日均
8	deposit_qravg	存款季日均
9	deposit_moavg	存款月日均
10	loan_bal	贷款余额
11	fina_bal	理财余额
12	bound_bal	国债余额
13	fund_bal	基金余额
14	insu_bal	保险余额
15	salary_card_flag	是否代发工资
16	sign_ebank_flag	是否为网银客户
17	credit_card_flag	是否为信用卡客户
	…	……

3）交易数据表如表 11-4 所示。

表 11-4　交易数据表

序　　号	变 量 名 称	变 量 标 签
1	mo_trnx_amt	月交易金额
2	mo_trnx_cnt	月交易次数
3	mo_draw_amt	月取现金额
4	mo_draw_cnt	月取现金次数
5	mo_expd_amt	月消费金额
6	mo_expd_cnt	月消费次数
7	mo_trac_amt	月转账金额
8	mo_trac_cnt	月转账次数
9	mo_fcur_amt	月外币交易金额
10	mo_fcur_cnt	月外币交易次数
	…	……

4）风险数据表如表 11-5 所示。

表 11-5　风险数据表

序　　号	变 量 名 称	变 量 标 签
1	prop_times	放款次数
2	prop_amt	放款金额
3	overdue_amt	逾期金额
4	baddue_amt	呆滞金额
5	baddebt_amt	呆账金额
6	odue_times	逾期次数
7	long_odue_days	最长逾期天数
	…	……

营销结果数据通常为手工记录得到的 Excel 文档，需要进行批量导入，下面的 SAS 宏为 Excel 文件的 sheet 的批量导入程序。

```
%macro    excel_import (file,sheet,outlib,outpre,varnm=1,range=);
/*_____0.参数处理__________________________________________________________________*/
%if   %sysfunc(sum(%index(&file.,%str(%')),%index(&file.,%str(%"))))>=1
%then    %let file=%sysfunc(compress(&file.,%str(%"%')));
    %else    %let    file=%sysfunc(pathname(&file.));
%let    sheet=%lowcase(&sheet.);
%if    %bquote(&outlib.)^=    %then    %let    outlib=%lowcase(&outlib..);
%if %bquote(&outpre.)^=      %then    %let    outpre=%lowcase(&outpre.);
/*sheet 变量拆分*/
%local    sheet_ex;
%let    sheet_ex=%scan(%bquote(&sheet.),2,|);
```

```
%let    sheet=%scan(%bquote(&sheet.),1,|);
/*sheet_list 变量列表*/
%local    sheet_list;
%if    %bquote(&sheet.)=%str(*) %then %do;
        libname    xlsfile    excel    "&file.";    /*Excel 指定引擎读取 xls 格式文件，xlsfile 指定 Excel 库*/
        proc    contents
             data=xlsfile._all_ directory
             memtype=data noprint
             out=mcr_xls_import_file nodetails;
         run;
        proc    sql;
         select    distinct compress(memname,' $')    into :sheet_list    separated    by    ' '
from    mcr_xls_import_file;
         quit;
        proc    datasets
             mt=data nolist;
         delete    mcr_xls_import_file;
         run;
        libname    xlsfile    clear;
%end;
%else    %let    sheet_list=&sheet.;
/*从 sheet_list、sheet_ex 列表拆分*/
%local i j;
%let i=1;
%do    %until(%scan(&sheet_list.,&i.,%str( ))= );
        %local    sheet_list_&i.;
        %let    sheet_list_&i.=%scan(&sheet_list.,&i.,%str( ));
        %let    i=%eval(&i.+1);
%end;
%let    j=1;
%do    %until(%scan(&sheet_ex.,&j.,%str( ))= );
        %local    sheet_ex_&j.;
        %let    sheet_ex_&j.=%scan(&sheet_ex.,&j.,%str( ));
        %let    j=%eval(&j.+1);
%end;

/*_____1.主程序___________________________________________________________________*/
%local    m n    sheet_nm sheet_tb    sheet_import_flag    out_sn;
%let    out_sn=1;
%do    m=1 %to %eval(&i.-1);
        %let    sheet_import_flag=1;
        /*判断 shee_nm 是不是在排除范围*/
        %let    sheet_nm=&&sheet_list_&m..$;
        %let    sheet_tb=%sysfunc(compress(&sheet_nm.,_,kad));
        %let    n=1;
        %do    %until(%bquote(&sheet_import_flag.)=0 or &n.=&j.);
```

```
%if %lowcase(&&sheet_list_&m..)=%lowcase(&&sheet_ex_&n..) %then %let
sheet_import_flag=0;
    %let n=%eval(&n.+1);
   %end;
   /*执行导入数据集操作*/
   %if &sheet_import_flag.=1 %then %do;
    Proc import out=%sysfunc(compress(&outlib.&outpre.&sheet_tb.))
               datafile= "&file."
               dbms=excel replace;
        range="&sheet_nm.&range."; /*range="sheet$a1:c9";*/ /*sheet="&sheet_tb.";*/
        %if %bquote(&varnm.)=1 %then %str(getnames=yes;);
%else %str(getnames=no;);
      mixed=yes;
       scantext=yes;
       usedate=yes;
     quit;
     %put <ok!>_&out_sn. &file.|&sheet_tb.[%sysfunc(coalescec(&range.,all))] →
&outlib.&outpre.&sheet_tb.;
     %let out_sn=%eval(&out_sn.+1);
   %end;
 %end;
 %mend excel_import;
```

【宏参数说明】

程序名称：%excel_import(file,sheet,outlib,outpre,varnm=1,range=);

功能说明：批量导入 Excel 的 sheet 内容。

参数 file：指定 Excel 文件的“物理路径”（需用引号，且包含文件扩展名）或 fileref，

如 filename stdata "c:\temp\state_data.xls"。

sheet：格式“需导入的 sheet<|需剔除的 sheet>”；各 sheet 名之间以空格分隔，*标识导入所有 sheet，如*|a b。

outlib：输出逻辑库，默认为系统默认库。

outpre：输出数据表名前缀，默认无。

varnm=1：定义是否使用 Excel 第一行作为变量名。默认 varnm=1，使用 Excel 第一行作为变量名；varnm=0 时，将 Excel 全部作为数据处理。

range=：定义读取 sheet 数据的范围，默认为空，读取所有数据。range=a1:c9 格式时读取相应范围 a1:c9 内的有效数据。

例：选取单个单元格 b2:b2，右下角单元格 e5:zzz99999（赋一个很大的值）。

输出 Excel 各 sheet 对应的数据集：&outlib..&outpre.sheet_name。

各数据集市变量准备结束后，将各主体数据集市的变量按照客户编号 user_id 和月份编号 month_nbr 进行横向合并。

```
proc sql;
   create table input_basic as select
```

```
      a.*, b.*, ……
     from input_basic_1 as a
      left  join  inpug_basic_2  as  b  on b.user_id=a.user_id and  b.month_nbr=a.month_nbr
      left  join
order  by user_id,month_nbr;
     quit;
```

生成关于基础变量的数据集 INPUT_BASIC，表头结构如表 11-6 所示。

表 11-6　表头结构

user_id	month_nbr	var_1	var_2	var_3	var_4	…

（4）变量衍生

基于 INPUT_BASIC 表中的基础变量衍生得到含衍生变量的表 INPUT_DERIVED。以其中一个变量 var 的衍生为例，SAS 代码如下：

```
data  input_derived;
     set  input_basic;
     format  var_1-var_12;
     by  user_id  month_nbr;
/*定义变量 var 对应的数组*/
     if f irst.user_id  then call missing(of var_1-var_12);
     array  var_ary[1:12] var_1-var_12;
     retain  var_1-var_12;
     var_ary[month_nbr]=var;
/*计算衍生变量*/
     var_max=max(of var_1-var_12);/*最大值*/
     var_min=min(of var_1-var_12);/*最小值*/
     avg_last_3mths=sum(of var_10-var_12)/3;/*最近 3 个月的平均值*/
/*输出*/
     if  last.user_id  then output;
     drop  var  month_nbr;
run;
```

（5）数据抽样

根据观察期进行排外：逾期次数超过 3 次、呆滞及呆账客户、账户状态异常客户等。

根据表现期进行排外：表现期销户的客户、表现期账户状态异常的客户等。

采用随机抽样，70%的样本用于建模，30%的样本用于样本内校验。抽样的程序如下：

```
data  input_train
      input_test;
     set  input_derived;
     random_nbr=uniform(123456);
     if  random_nbr>=0.3  then  output  input_train;
     if  random_nbr<0.3  then output  input_test;
run;
```

11.4.2 开发模型

1．变量处理

（1）变量细分组

变量细分组主要是对连续型变量进行处理，按照一定的步长将变量离散化。例如，存款余额可以按照每 500 元的步长进行离散化，并分别进行各分组的 WOE 值。

（2）变量粗分组

借助工具软件或 Excel 将 WOE 值相近的细分组进行合并，生成粗分组，并计算粗分组的 WOE 值。粗分组过程通常分组的 WOE 分布需满足单调性。如表 11-7 所示为性别变量的粗分组结果。

表 11-7　性别变量粗分组

性　别	好	坏	%好	%坏	woe_train	iv_train
缺失	615	515	5.98%	3.56%	0.5198	0.013
女	4603	7056	44.75%	48.76%	−0.0857	0.003
男	5068	6901	49.27%	47.68%	0.0326	0.001
合计	10286	14472				0.017
缺失	268	219	6.24%	3.50%	0.5779	0.016
女	1907	3003	44.39%	48.00%	−0.0781	0.003
男	2121	3035	49.37%	48.50%	0.0178	0
合计	4296	6257				0.019

变量信息值如表 11-8 所示。

表 11-8　变量信息值

变 量 名 称	变 量 标 签	iv_train	iv_test
deposit_bal	存款余额	0.29207	0.25105
sex	性别	0.01655	0.0188

通常选择 IV 值较高的前 50 个变量进入模型。

2．模型建立

将变量的 WOE 和权重作为模型输入，采用 LOGISTIC 回归，使用逐步回归的方法对变量进行筛选，最终得到一组模型变量及回归系数。回归结果如表 11-9 所示。回归程序如下：

```
proc    logistic    data=input_train;
        model    response(event='1')=var1-var50
                     /selection=STEPWISE
                 details
                 outroc=ROC;
```

```
    weight WEIGHT;
run;
```

表 11-9　回归结果

Parameter	Estimate	Standard error	Wald Chi-Square	Pr > ChiSq
var1	1.6499	0.1428	133.434	<.0001
var3	0.3905	0.0242	259.78	<.0001
var5	0.7262	0.0914	63.1509	<.0001

3．评分转换

评分的分值可以根据下面的公式计算，最后生成评分卡：

$$Score = 28.8539 \times \left(\sum_{i} a \times woe_i + b \right) + 513.561$$

以年龄变量为例，评分值如表 11-10 所示。

表 11-10　评分值

变 量 名 称	特　征　值	分　　值
age（年龄）	缺失	48
	[21,26)	57
	[26,35)	60
	[35,50)	63
	[50,max]	59

4．模型评估

模型的 K-S 值为 33.21，对应的评分为 520 分，如图 11-12 所示。

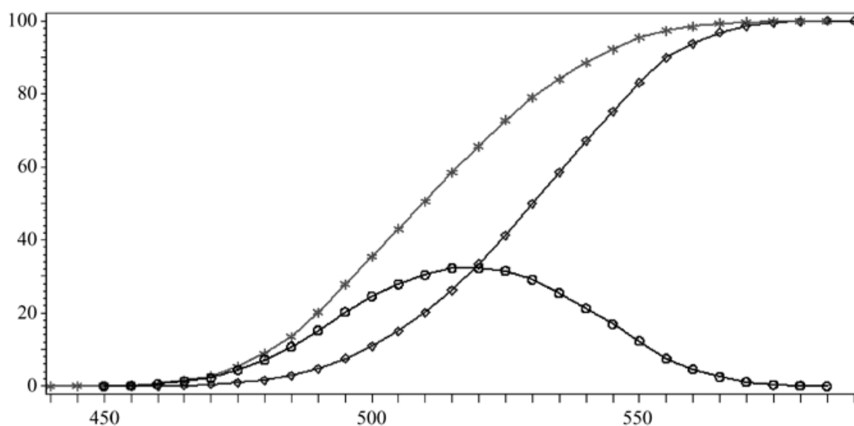


图 11-12　K-S 值

模型的分离度指标为 0.61，如图 11-13 所示。

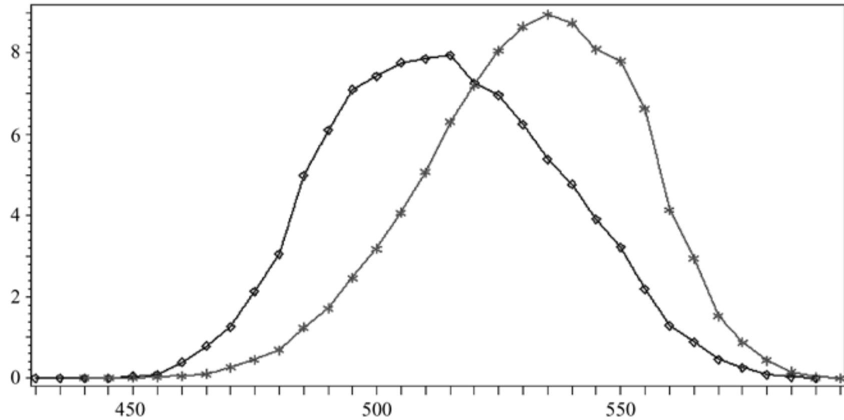


图 11-13 分离度

11.4.3 模型应用

模型开发完成以后，应用 INPUT_TEST 数据集对模型进行样本内校验，由于缺乏样本外数据，故省略样本外校验。通过样本内校验比较 K-S 和分离度指标变化不大，认为模型稳定。

模型验证确认以后，基于模型开发产品营销策略形成决策树，生成策略结果。由于该模型针对某特定产品营销，不存在周期策略问题，仅需生成一次策略结果供业务使用。在业务执行完成后，根据营销效果需对模型进行再次评价和经验总结。

第 12 章　高级应用技巧

12.1　自动变量与临时变量应用

12.1.1　自动变量_N_与_ERROR_应用

PDV（Program Data Vector）是 SAS 在建立数据集时在内存开辟的一块逻辑区域。当建立数据集代码执行时，SAS 从输入缓冲区读入观测到 PDV 中或通过语句在 PDV 中生成观测，并在 PDV 中对变量进行其他操作。当读取观测到 PDV 时每次仅读入一条记录，SAS 会在 PDV 中为读入观测变量分配空间，并把读入记录的具体数值赋给变量，当新的观测读入时变量的数值相应的会被更新，所有操作完成后观测被输出到结果数据集中，在无指定操作时变量的数值不会被清除。

自动变量是 SAS 在执行数据操作时在 PDV 中自动生成的变量，并不会输出到结果中去。自动变量记录了数据处理的相关信息，记录的信息非常有价值，可以通过自动变量来了解 SAS 进行数据处理的过程或利用其进行其他复杂的数据操作。

SAS 在执行 DATA 步过程时，在 PDV 中生成自动变量_N_ 和 _ERROR_。_N_初始值为 1，在每次读入数据观测时，数值自动加 1，由此可见_N_代表 SAS 执行 DATA 步过程中重复执行数据读取的次数，即读入 PDV 中观测的条数。_ERROR_默认值为 0，当数据处理过程中遇到错误时，值被改写为 1。常见的错误类型有数据输入错误、数据转换错误、数据格式不匹配、分母为 0 等。SAS 在遇到错误时，会将具体的错误信息输出到 LOG 中，可以在 LOG 中利用_N_与_ERROR_迅速定位错误所在的位置并查找错误产生的原因。

【例 12.1】 通过 cards 语句输入新增金额 newly 以及剩余金额 remain 的数值，并计算新增金额占比。

```
/*_ERROR_的使用*/
data  newly_pct;
      input  newly  remain;
        rate=newly/remain;
        v_error=_error_;    *_ERROR_的数值;
        if v_error=1 then message="error";
      cards;
      1000  10000
      2000  50000
      3000  0
      4000  10000
      5000  x
      ;
run;
```

【程序解读】

1）通过 cards 语句读入变量 newly 及 remain 对应的数据。

2）计算新增占比 rate。

3）将_ERROR_赋值给变量 v_ERROR。

数据步结果显示如表 12-1 所示。

表 12-1　数据步处理结果显示

newly	remain	rate	v_ERROR	message
1000	10000	0.1	0	
2000	50000	0.04	0	
3000	0		1	ERROR
4000	10000	0.4	0	
5000			1	ERROR

日志：

```
15     /*_ERROR_的使用*/
16     data   newly_pct;
17            input   newly   remain;
18            rate=newly/remain;
19            v_error=_error_;*_error_的数值;
20            if v_error=1 then message="error";
21            cards;

NOTE: 检测到 0 为除数，位置: 行 18 列 15。
RULE:
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+----0
24                    3000   0
newly=3000 remain=0 rate=. v_error=1 message=error _ERROR_=1 _N_=3
NOTE：在第 26 行、第 11～11 列中有对“remain”无效的数据。
26                    5000   x
newly=5000 remain=. rate=. v_error=1 message=error _ERROR_=1 _N_=5
NOTE：缺失值的生成是对缺失值执行操作的结果。
      指定每个位置的方式:（次数）（行:列）。
      1，位置：18:15。
NOTE：在以下位置无法执行算术运算。运算结果已设为缺失值。
      每个位置的指定方式:（次数）（行:列）。
      1，位置: 18:15。
NOTE：数据集 WORK.NEWLY_PCT 有 5 个观测和 5 个变量。
NOTE：“DATA 语句”所用时间（总处理时间）:
      实际时间            0.01 秒
      CPU 时间            0.01 秒


27            ;
28     run;14      run;
```

【日志解读】

1）_N_=3 时，_ERROR_=1，即第三条观测出现错误，此观测中 remain=0 无效，因为 0 不能作为分母。

2）_N_=5 时，_ERROR_=1，即第五条观测出现错误，此观测中 remain 的输入值有误，输入值为 x，非数值型输入数据。

【例 12.2】 对例 12.1 进行改造，在数据集 newly_pct 中筛选新增金额大于 2000 且小于 5000 的记录。

方法一：

```
Data   gt_2000_1;
       Set   newly_pct;
        Where   2000<newly<5000;
        id=_n_;
        keep   id   newly   remain;
run;
```

gt_2000_1 结果显示如表 12-2 所示。

表 12-2　gt_2000_1 对应结果

newly	remain	id
3000	0	1
4000	10000	2

方法二：

```
data   gt_2000_2;
        set   newly_pct;
        if   2000<newly<5000;
        id=_n_;
        keep   id   newly   remain;
run;
```

gt_2000_2 显示结果如表 12-3 所示。

表 12-3　gt_2000_2 对应结果

newly	remain	id
3000	0	3
4000	10000	4

通过以上两种方法的结果对比可以看出，变量 id 的具体数值并不一致，原因在于 where 语句在将数据读入 PDV 之前执行，而 if 语句在 PDV 中执行。方法一中的 where 语句先判断 newly 是否大于 2000 且小于 5000，然后将观测读入 PDV 中，符合条件的第一条观测 id 为 1，同理第二条观测 id 为 2。方法二中 if 语句是在数据读入 PDV 之后执行，第三条和第四条观测符合条件，而其 id 数值分别为 3 和 4。

12.1.2　临时变量FIRST.变量与LAST.变量的应用

SAS 执行 SORT 过程时，在按变量排序后的结果中会生成临时变量 FIRST.变量与 LAST.变量，每一个排序变量（BY 变量）都会生成这样两个临时变量。FIRST.变量记录在每一个 BY 组中，观测是否是第一条观测；而 LAST.变量记录在每一个 BY 组中，观测是否是最后一条观测。其中，BY 组是排序变量的任一个值的所有观测构成的组，如果有多个排序变量，BY 组是当前变量在之前变量取值下的任一值的所有观测构成的组。如果观测为 BY 组中的第一条观测，则 FIRST.变量的值为 1，否则为 0；如果观测为 BY 组的最后一条观测，则 LAST.变量的值为 1，否则为 0。

【例 12.3】 某小组员工入职后每月的销售额记录 sales，请找出每位员工首月的销售额。其中 id 为员工编号，year 为年份，month 为月份，sales 为销售额。

数据集 sales 数据信息如表 12-4 所示。

表 12-4　数据集 sales 数据信息

id	year	month	sales
1001	2011	9	55
1001	2011	10	66
1001	2011	11	69
1001	2011	12	98
1001	2012	1	96
1002	2011	9	69
1002	2011	10	72
1003	2011	9	58
1003	2011	12	79
1003	2012	1	76
1004	2011	10	59
1004	2011	11	58
1005	2011	11	99

首先，通过下面的程序来认知 FIRST.变量与 LAST.变量。

```
proc   sort   data=sales;
       by   id   year   month;
run;
data   sales_info;
        set   sales;
        by   id   year   month;
        id_first=first.id;
        id_last=last.id;
        year_first=first.year;
        year_last=last.year;
run;
```

【程序解读】

1）通过调用 SORT 过程对数据集 sales 按照员工编号、年份、月份排序。

2）将 first.id 值赋给变量 id_first，将 last.id 值赋给 id_last。

3）将 first.year 值赋给变量 year_first，将 last.year 值赋给 year_last。

结果显示的 sales_info 信息如表 12-5 所示。

表 12-5　sales_info 数据集信息显示

id	year	month	sales	id_first	id_last	year_first	year_last
1001	2011	9	55	1	0	1	0
1001	2011	10	66	0	0	0	0
1001	2011	11	69	0	0	0	0
1001	2011	12	98	0	0	0	1
1001	2012	1	96	0	1	1	1
1002	2011	9	69	1	0	1	0
1002	2011	10	72	0	1	0	1
1003	2011	9	58	1	0	1	0
1003	2011	12	79	0	0	0	1
1003	2012	1	76	0	1	1	1
1004	2011	10	59	1	0	1	0
1004	2011	11	58	0	1	0	1
1005	2011	11	99	1	1	1	1

由结果数据可以看出，id_first 和 id_last 标示了观测是否是 id 所属观测组中的第一条观测与最后一条观测。而 year_first 和 year_last 标示了在 id 当前取值下，观测是否是 year 所属观测组中的第一条观测与最后一条观测。

如本案例中结果中的第五条观测，观测为 id=1001 下所对应有观测的最后一条记录，故 id_last=1；同时，观测为 id=1001 且 year=2012 下所有观测的唯一一条记录，故 year_first=1 且 year_last=1。

【例 12.4】 对例 12.3 进行改造，找出每位员工首条记录的销售额。

```
Proc   sort   data=sales;
        by   id   year   month;
run;
data   first_sales;
        set   sales;
        by   id   year   month;
        if   first.id   then   output;
run;
```

【程序解读】

1）SORT 过程将每位员工的销售额按年份、月份排序，默认为升序。

2）如果 id_first 等于 1 即观测为员工的首条观测，则输出结果。

数据集 first_sale 结果显示如表 12-6 所示。

表 12-6　first_sale 数据集数据显示

id	year	month	sales
1001	2011	9	55
1002	2011	9	69
1003	2011	9	58
1004	2011	10	59
1005	2011	11	99

【例 12.5】 对例 12.3 进行改造，求每位员工的总销售额。

```
Proc  sort  data=sales;
       by  id  year  month;
run;
data  sales_sum;
       set  sales;
       by  id  year  month;
      retain  sales_sum; *记录总销售额;
/*if-else 语句实现累加效果*/
      If  first.id  then  sales_sum=sales;
      else  sales_sum=sales_sum+sales;
/*输出每个 ID 的最后一条记录，sales_sum 即为总销售额*/
      if  last.id  then  output;
      keep  id  sales_sum;
run;
```

【程序解读】

1）对数据集 sales 按照员工编号、年份、月份排序。

2）用 retain 生成新的变量 sales_sum 来记录总销售额。

3）if-else 语句实现累加效果。

4）将每位员工的最后一条记录输出，sales_sum 为总销售额。

数据集 sales_sum 数据显示如表 12-7 所示。

表 12-7　数据集 sales_sum 数据信息显示

id	sales_sum
1001	384
1002	141
1003	213
1004	117
1005	99

12.2　SAS索引应用

12.2.1　索引简介

1．索引的定义

索引是 SAS 为了方便直接访问数据集的特定观测而生成一个附加性文件，并不是数据集的必须组成部分。索引由单个变量或多个变量组成，默认按升序次序记录变量值，并记录变量值对应观测所在的位置。换句话说，索引具有通过变量值定位观测所在位置的功能。

索引好比是一本书的目录，可以帮助读者快速定位特定章节所在的页数。在访问数据时，利用索引可以很好地获取特定观测所在的位置，提高数据访问速度和程序的执行效率。假如想从一个记录了公司 10000 名员工的信息的数据集中找出编号为 00898 的员工的信息，在没有索引的情况下，SAS 会按数据存储的顺序逐条读取观测，判断观测中员工号是否是 00898，直到所有的观测访问完毕最后将符合条件的结果输出，每条观测都会被读取一次。如果存在建立在员工号码上的索引，通过索引可以直接找到员工号等于 00898 的所有观测所在的位置，直接读取数据输出结果，并不需要读取数据集中所有的观测。

SAS 可以在生成数据集的时候建立索引，也可以为已经存在的数据集建立索引。针对每一个数据集，可以建立一个索引或者多个索引。索引类型分为简单索引或复合索引。如果索引存在，SAS 会把它当做数据集的一部分。当观测增加或删减以及变量值改变时，索引会自动更新。

2．索引的优点

索引记录了变量值所属观测所在的位置，在以下情况下使用索引可以提高数据访问速度：

1）索引在执行 Where 语句读取数据集时，会提供快速高效的数据访问速度。需要注意的是，并不是每次执行 Where 语句时都会使用索引，SAS 默认情况下会比较使用索引访问和直接访问数据集的时间来判断是否使用索引。

例如，从 1 万个 ID 的 100 万条记录中查找一个 ID 的信息，存在建立在 ID 上的索引时，查询时间为 0.2s，而不存在建立在 ID 上的索引时查询时间为 1s。对于小的数据集，索引的存在对数据访问速度并没有太大的提升，但是对于从一个很大的数据集中抽取极少的观测时，索引的使用会大大减少数据读取时间。

2）索引在执行 BY 语句时，会使结果按照索引中变量值的顺序返回观测，不管数据集中存储观测的顺序如何。在使用 SORT 步时，如果对索引变量排序，默认情况下索引并不会被使用，并且程序会报错。

【例 12.6】 索引对 BY 语句的影响。

```
/*在 id 上建立索引*/
data   testfile (index=(id));
       input   id   $   amt;
       cards;
       101 50
       109 100
       112 30
       102 69
       103 88
       120 36
       ;
run;
data   test_by_idx;
        set   testfile;
        by   id;
run;
```

程序执行后结果显示数据信息如表 12-8 所示

表 12-8　test_bu_idx 数据集信息显示

id	amt
101	50
102	69
103	88
109	100
112	30
120	36

【结果解读】

因为存在建立在变量 id 上的索引，并且程序中的 BY 语句引用了索引变量，故结果按照变量 id 升序次序（索引中 id 的顺序）返回观测。

3）索引在使用 SQL 拼接数据集时，会提高数据查询效率。当使用 SQL 来拼接数据时，可以根据索引变量来直接匹配观测。

索引虽然可以减少定位观测的时间，以提高数据集抽取的速度，特别是对于从很大的数据集中抽取少量观测时效果非常明显，但是索引的生成、存储以及维护都存在成本。当不确定是否需要生成索引时，必须要考虑由此带来的资源消耗和效率提升之间的平衡。

3．认识索引文件

索引文件是与 SAS 数据文件相关联并具有相同文件名称的 SAS 索引型文件，每个数据文件只有一个索引文件，所有的索引都存放在唯一的索引文件中。

在索引文件中，索引变量的每一个值对应着一个或多个记录标示符，记录标示符代表着变量值所属观测在数据文件中的位置。表 12-9 所示为索引文件的形式。

表 12-9　索引文件的形式

变　量　值	记录标示符
111	1,26
222	2,5,49
333	3,51
444	6,9,25,95
555	4,12
666	7,8,85
…	…

当 SAS 处理数据查询时，如执行一个 where 语句，首先使用二分法在索引文件中查找定位符合条件的变量值，然后根据变量值对应的记录标示符读取相应的观测。如果变量值有多个记录标示符，SAS 按照记录标示符的顺序读取观测。

索引分为简单索引和复合索引。在创建索引时，需要指定索引名称和创建索引的变量，通常创建索引的变量被称为主键，不能为同一数据集创建同名索引。

简单索引是最常见的索引。它是由单一变量生成的索引，变量可以为数值型或字符型。当创建简单索引时，SAS 默认用变量的名称作为索引的名称。

复合索引是由两个或两个以上变量联合生成的索引，变量可以是数值型、字符型或两者混合使用。当创建复合索引时，需要为索引指定唯一的索引名称。

索引类型的选择，主要根据数据访问的具体形式来确定。如果经常使用某个主键来访问数据，应该建立单一索引，而经常使用多个主键来访问数据时需要建立复合索引。

12.2.2　索引的创建与删除

1．索引的创建

在 DATA 步、PROC 步和 SQL 过程中都可以创建索引。创建索引的方法主要有以下几种：

（1）DATA 步中使用 INDEX=选项创建索引

使用 INDEX=选项创建索引的一般形式如下：

```
SAS-data-file-name(INDEX=(
index-specification-1</option>…index-specification-n</option>)
        index-specification 的语法如下：
        simple    index : the name of the key variable
        composite    index : index-name(list of key variables)
```

其中，尖括号内为选项语句。选项语句中经常使用的关键字有 UNIQUE 与 NOMISS。UNIQUE 要求创建索引的变量的数值必须是唯一的，不允许重复值的存在，否则程序会报错，索引也不会被创建。NOMISS 表示索引中不会包含变量的缺失值，如果变量值为缺失，则索引不能直接获取所属观测所在的位置。

【例 12.7】 创建索引并输出索引创建信息。

```
/*创建单一索引 id、复合索引 id_type*/
Data    idx_data(index=(id    id_type=(id    type)));
          Set    id_amt;;
run;
/*将索引信息输出*/
proc    contents    data=idx_data;
quit;
```

输出的索引信息如图 12-1 所示。

按字母排序的索引和属性列表

#	索引	唯一值个数	变量
1	id	6	
2	id_type	7	id type

图 12-1　查看 idx_data 索引信息

（2）使用 PROC　DATASETS 创建索引

DATASETS 步创建索引的一般形式如下：

```
PROC    DATASETS    LIBRARY=libref;
```

```
        MODIFY   SAS-data-file-name;
        INDEX   CREATE    index-specification-n</option>;
QUIT;
```

参考代码如下：

```
proc    datasets   lib=work;
      modify   testfile;
      index   create   type/nomiss; *单一索引;
      index   create   id_type=(id   type)/unique; *复合索引;
quit;
```

（3）使用 PROC　SQL 创建索引

PROC　SQL 创建索引的一般形式如下：

```
PROC   SQL;
        CREATE   INDEX   index-name   ON   table-name(columm-name-1,……,columm-name-n);
QUIT;
```

参考代码如下：

```
proc sql;
      create   index id   on test_file(id);
      create index   id_type   on   test_file(id,type);
quit;
```

2．索引的删除

在 DATA 步、PROC 步和 SQL 中都可以删除索引。删除索引的方法主要有以下几种：

（1）DATA 步删除索引

当使用 DATA 步创建同名数据集时，原有的索引会被删除。

参考代码如下：

```
data   test;
      set   test;
run;
```

（2）使用 PROC　DATASETS 删除索引

DATASETS 步删除索引与创建索引相似，形式如下：

```
PROC   DATASETS   LIBRARY=libref;
        MODIFY   SAS-data-file-name;
        INDEX   DELETE   index-name
QUIT;
```

参考代码如下：

```
proc   datasets   lib=work;
      modify   testfile;
      index   delete   type   id_type;
```

```
quit;
```

（3）使用 PROC　SQL 删除索引

PROC　SQL 删除索引的语句形式如下：

```
PROC   SQL;
       DROP   INDEX   index-name   FROM   table-name;
QUIT;
```

参考代码如下：

```
proc   sql;
       drop   index   id   from   test_file;
quit;
```

12.2.3　索引的应用

索引的使用会减少数据访问的时间以提高程序的运行效率。在默认情况下，SAS 在执行代码时首先会判断表达式是否可引用已存在的索引，并在可引用的索引中选择最优的索引，然后比较使用索引和不使用索引读取数据的时间，最后确定是否使用索引来读取数据。在编写代码时，要使用合适的表达式语句，让 SAS 来识别可使用的索引。

SAS 执行 where 语句时，会首先判断 where 语句中是否包含单一索引中的主键或复合索引的首变量，确定可引用的索引，然后在可引用的索引中选择满足条件最多且提供观测数量最少的索引。where 语句可能包含多个条件，每个条件中可能包含多个索引变量，但是 SAS 最终在可引用的索引中会选择一个最优的索引作为使用对象。

【例 12.8】 认知 SAS 在可引用的索引中选择最优索引。

```
data   score (index=(team id));
       input   team   $   id   $   score;
       cards;
       a 101 99
       a 102 86
       b 103 68
       b 104 63
       b 105 89
       c 101 73
       ;
run;
options   msglevel=i;
proc   print   data=score;
       where   team='a'   and   id='102';
run;
proc   print   data=score;
       where   team='c'   and   id='101';
run;
```

日志显示如图 12−2 所示。

```
295  data score(index=(team id));
296      input team $ id $ score;
297      cards;

NOTE: 数据集 WORK.SCORE 有 6 个观测和 3 个变量。
NOTE: 已定义简单索引 id。
NOTE: 已定义简单索引 team。
NOTE: “DATA 语句” 所用时间（总处理时间）:
      实际时间          0.01 秒
      CPU 时间          0.01 秒


304      ;
305  run;
306  options msglevel=i;
307  proc print data=score;
308      where team='a' and id='102';
INFO: 选择了索引 id 用于 WHERE 子句优化
309  run;

NOTE: 从数据集 WORK.SCORE. 读取了 1 个观测
      WHERE (team='a') and (id='102');
NOTE: “PROCEDURE PRINT” 所用时间（总处理时间）:
      实际时间          0.01 秒
      CPU 时间          0.01 秒


310  proc print data=score;
311      where team='c' and id='101';
INFO: 选择了索引 team 用于 WHERE 子句优化
312  run;

NOTE: 从数据集 WORK.SCORE. 读取了 1 个观测
      WHERE (team='c') and (id='101');
NOTE: “PROCEDURE PRINT” 所用时间（总处理时间）:
      实际时间          0.00 秒
      CPU 时间          0.00 秒
```

图 12-2　日志窗口显示信息

【日志解读】

1）为数据集 score 创建单一索引 team 和 id。

2）PRINT 过程中，在执行 where team='a' and id='102'时，team='a'有两条观测，而 id='102'有一条观测，SAS 最终选择索引 id 来读取数据；

3）PRINT 过程中，在执行 where team='c' and id='101'时，team='a'有一条观测而 id='102'有两条观测，SAS 最终选择索引 team 来读取数据。

通过比较使用索引和不使用索引读取数据的时间后，SAS 可能选择不使用索引来处理数据。默认情况时，SAS 自行选择 where 语句是否使用索引。在编写代码时，可以使用 IDXWHERE=选项来强制 where 语句中使用可引用的索引。当使用 IDXWHERE=YES 选项时，SAS 强制 where 语句使用索引；而使用 IDXWHERE=NO 选项时，SAS 强制 where 语句不使用索引。

【例 12.9】 IDXWHERE=选项的使用。

```
data   score(index=(id));
       input   id   $   score;
       cards;
       101 99
       102 86
       103 68
       104 63
       105 61
```

```
        106 85
        107 50
        108 71
        109 95
        ;
run;
options    msglevel=i;
/*强制使用索引*/
Proc    print    data=score (idxwhere=yes);
        where    id    in ('103','105','107');
        title    "With idxwhere=yes";
run;
/*强制不使用索引*/
proc    print    data=score (idxwhere=no);
        where    id    in ('103','105','107');
        title    "With idxwhere=no";
run;
/*SAS 自行选择是否使用索引*/
proc    print    data=score;
        where    id    in ('103','105','107');
        title "default";
run;
```

日志显示如图 12-3 所示。

```
223  options msglevel=i;
224  /*强制使用索引*/
225  proc print data=score(idxwhere=yes);
226      where id in ('103','105','107');
INFO: 数据集选项 (IDXWHERE=YES) 强制 Where 子句处理使用索引，而非顺序传递。
INFO: 选择了索引 id 用于 WHERE 子句优化
227      title "With idxwhere=yes";
228  run;

NOTE: 从数据集 WORK.SCORE. 读取了 3 个观测
      WHERE id in ('103', '105', '107');
NOTE: “PROCEDURE PRINT”所用时间(总处理时间):
      实际时间           0.01 秒
      CPU 时间           0.01 秒


229  /*强制不使用索引*/
230  proc print data=score(idxwhere=no);
231      where id in ('103','105','107');
INFO: 数据集选项 (IDXWHERE=NO) 强制 Where 子句处理使用数据的顺序传递，而非索引。
232      title "With idxwhere=no";
233  run;

NOTE: 从数据集 WORK.SCORE. 读取了 3 个观测
      WHERE id in ('103', '105', '107');
NOTE: “PROCEDURE PRINT”所用时间(总处理时间):
      实际时间           0.00 秒
      CPU 时间           0.00 秒


234  /*SAS自行选择是否使用索引*/
235  proc print data=score;
236      where id in ('103','105','107');
INFO: 选择了索引 id 用于 WHERE 子句优化
237      title "default";
238  run;

NOTE: 从数据集 WORK.SCORE. 读取了 3 个观测
      WHERE id in ('103', '105', '107');
NOTE: “PROCEDURE PRINT”所用时间(总处理时间):
      实际时间           0.00 秒
      CPU 时间           0.00 秒
```

图 12-3　日志显示信息

【日志解读】

1）IDXWHERE=YES 时，SAS 强制 where 语句使用索引。

2）IDXWHERE=NO 时，SAS 强制 where 语句不使用索引。

3）默认情况时，SAS 自行选择 where 语句是否使用索引。示例中 SAS 选择了使用索引 ID 来优化查询。

在 where 语句中，以下条件表达式形式会引用存在的索引。

① 比较运算符、IN 操作符：

```
where   age=25;
where   score   gt   85;
where   id   in ('201','302');
```

② NOT 操作符：

```
Where   id   not   in ('501','902');
```

③ 特定操作符，如 CONTAINS、LIKE、IS MULL、BETWEEN…AND 等。

```
where   name   contains   'Jim';
where   age   between   25   and   28;
where   name   like   "%Tom_";
```

④ 特定函数 TRIM 与 SUBSTR，使用 SUBSTR 函数时需要从首位抽取并且抽取长度要小于等于变量的长度。

```
where   trim(lastname)='Jerry';
where   substr(firstname,1,1)='J';
```

SAS 程序中不使用索引的情况如下：

1）DATA 步中的 IF 语句。

2）SAS 判断需要读取所有观测才能满足 where 条件时。

3）SAS 判断读取所有观测比使用索引访问数据速度更快。

4）不存在任何索引满足数据查询。

5）where 语句中不规范的表达式。

6）where 语句中使用除 TRIM 与 SUBSTR 之外的函数。

12.3 自定义FORMAT格式应用

FORMAT 可以格式化数值或字符的输出格式，使输出结果更易于阅读或便于统计，FORMAT 也可以指定变量以特定格式参与运算。SAS 自带非常丰富的数值型、日期型及字符型格式，用户也可以自定义 FORMAT 格式并保存在永久数据库中以备下次使用。用户自定义的 FORMAT 格式，在形式上具有灵活性，在内容上具有丰富性，在数据处理时可以关联数据或筛选数据，减少 SORT 与 MERGE 的使用，提高程序的运行效率。

【例 12.10】 假设目前存在两个数据集，其中一个记录了销售级别及对应的佣金比例，

另一个记录了上半年每位员工的销售级别及销售金额，计算每位员工的佣金金额。

```
/*佣金比例*/
Data   grade_rate;
       Input   grade $   rate;
       cards;
       A 0.10
       B 0.13
       C 0.15
       D 0.18
       E 0.20
       F 0.25
       ;
run;

/*销售金额*/
Data   sales_amt;
       input   id   $   grade   $   sales_amt;
       cards;
       101 A 110
       102 B 102
       103 C 132
       104 D 114
       105 E 150
       106 F 168
       111 A 160
       112 B 122
       123 C 138
       124 D 125
       108 E 156
       109 F 118
       ;
run;

/*建立 FORMAT*/
proc   sort   data=grade_rate   nodupkey;
       by   grade;
run;
data   fmt_rate;
       set   grade_rate;
       start=grade;
       label=rate;
       type='c';
       fmtname='grd_rate';
       keep   start   label   type   fmtname;
run;
proc   format   cntlin=fmt_rate;
```

```
run;

/*计算佣金金额*/
data   cms_amt;
       set   sales_amt;
       rate=put(grade,$grd_rate.);
       cms_amt=sales_amt*rate;
run;
```

【程序解读】

1）通过 PROC FORMAT 将佣金比例数据集定义为 FORMAT 格式，相当于创建 GRADE 与 RATE 的对应关系。需要注意的是，变量 GRADE 不能有重复值。

2）通过 PUT 函数在 FORMAT 格式中找到 GRADE 对应的 RATE。

3）计算佣金金额。

结果如表 12-10 所示。

表 12-10　cms_amt 数据集信息

id	grade	sales_amt	rate	cms_amt
101	A	110	0.1	11
102	B	102	0.13	13.26
103	C	132	0.15	19.8
104	D	114	0.18	20.52
105	E	150	0.2	30
106	F	168	0.25	42
111	A	160	0.1	16
112	B	122	0.13	15.86
123	C	138	0.15	20.7
124	D	125	0.18	22.5
108	E	156	0.2	31.2
109	F	118	0.25	29.5

上例中将销售级别对应佣金比例数据集定义为 FORMAT 格式，在 DATA 步中调用格式来获取佣金比例并计算佣金。代码中并没有将两个数据集首先排序然后拼接，减少了排序数据集消耗的时间。

【例 12.11】 接上例，查找佣金比例为 0.2 的员工。

```
/*建立 FORMAT*/
Proc   sort   data=grade_rate nodupkey;
       by   grade;
run;
data   grade_rate;
       set   grade_rate;
       start=grade;
```

```
        label=rate;
        type='i';
        fmtname='grade_rate';
        keep  start  label  type  fmtname;
run;
proc  format  cntlin=grade_rate;
run;

/*筛选佣金比例等于 0.2 的员工*/
Data  id_sub;
      set  sales_amt;
      where  input(grade,grade_rate.)=0.2;
run;
```

【程序解读】

1）首先创建 FORMAT 格式，TYPE 选项与上例中的并不一致，本例中为 I，原因在于本例中 FORMAT 格式需要用在 where 语句的 INPUT 函数中。关于 TYPE 选项的详细使用情景可查阅 SAS 帮助。

2）通过 where 语句判断条件直接抽取数据。

通过以上两个例子可以看出，在数据查询过程中，自定义 FORMAT 格式的应用可以实现关联信息或筛选数据的作用，避免数据拼接过程中多次使用 SORT 与 MERGE，在对比较大的数据集操作时，可以大大减少程序运行时间提高数据处理效率。在实际操作时，经常将参数表或者小数据集定义为 FORMAT 格式，在对其他数据集操作时引用自定义的 FORMAT 格式。

12.4　HASH对象的应用

在 DATA 步中使用 HASH 对象，不但可以快速有效地检索和读取数据，而且可以实现多个数据集之间的拼接，提高数据查询的效率。HASH 对象相当于一张行列表，在内存中存储与更新，由关键字变量（称为主键）和信息变量两部分组成。在 DATA 步过程中可以直接通过 HASH 对象的主键来匹配观测，也可以将 HASH 对象中的信息变量输出到结果中去，在 DATA 步结束后 HASH 对象失效。下面来简单的认识一下如何定义一个 HASH 对象。

定义 HASH 对象主要由以下步骤组成：

1）声明 HASH 对象。相关的语法如下：

```
declare  hash  myhash;
myhash = _new_ hash();
```

其中，declare 为关键字，myhash 为定义的 HASH 对象的名称，用户可根据实际情况来命名 HASH 对象。或者简化为

```
declare  hash  myhash();
```

2）初始化 HASH 对象。相关的语法如下：

```
Declare  hash  object_name(argument_tag-1 : value-1
```

```
        <, ...argument_tag-n: value-n>);
OR
object_name = _new_ hash(argument_tag-1: value-1
      <, ...argument_tag-n: value-n>);
```

其中经常使用的参数标签如下：

① 指定作为 HASH 对象的数据集的名称，语法如下：

```
dataset: 'dataset_name'
```

② 是否忽略主键中重复值的存在，语法如下：

```
duplicate: 'option'
```

默认情况下只存储重复数值中的首条记录，忽略重复数值的后续记录。若使用'replace' | 'r'表示只存储重复数值中的最后一条记录，忽略之前的重复值记录；而'error' | 'e'表示如果遇到重复数值则报错。

③ 是否对主键排序。语法如下：

```
ordered: 'option'
```

默认情况下，对主键并不做排序处理。选项为 YES' | 'Y'表示对主键进行排序，默认为升序顺序；'NO' | 'N'表示不排序。'ascending' | 'a'表示按照升序顺序排序，而'descending' | 'd'表示按照降序顺序排序。

3）指定 HASH 对象的主键和信息变量，并结束 HASH 对象的初始化。相关语法如下：

```
myhash.definekey('key');
myhash.definedata('data');
myhash.definedone();
```

其中，myhash.definekey('key')为定义 HASH 对象的主键。主键可以为单个变量也可以由多个变量组成。myhash.definedata('data')为定义信息变量。信息变量不能为空，可以是主键也可以为其他多个变量。需要注意的是，在指定主键和信息变量时，需要首先定义变量的长度。

此过程中调用了 HASH 对象的方法，常用的方法除 DEFINEKEY 、DEFINEDATA 和 DEFINEDONE 外，还有 FIND。DEFINEKEY 用来指定 HASH 对象的主键，DEFINEDATA 用来指定 HASH 对象的信息变量，DEFINEDONE 用来结束 HASH 对象的初始化。而 FIND 用来检索某一数值是否出现在主键中，若检索到则返回数值 0，没有检索到则返回一个非零值，可以利用返回值来匹配观测，实现 MERGE 的作用。

【例 12.12】 使用 HASH 对象来筛选数据。已有新入职员工信息表，在所有员工销售额表中筛选出新入职员工的销售额。

```
/*新入职员工信息*/
Data   id_newly;
      input   id $   epl_ym;
      cards;
      1101 201201
```

```
        1102 201201
        1123 201203
        1105 201202
        1104 201202
        1115 201202
        ;
run;

/*每位员工的销售额及销售级别*/
data   sales_all;
        input   id   $   grade $   amt;
        cards;
        1001 A 561
        1101 C 256
        1002 B 421
        1003 A 691
        1005 A 555
        1004 B 398
        1015 A 402
        1102 C 128
        1123 D 96
        1105 C 196
        1104 D 89
        1086 B 632
        1093 A 701
        1115 C 221
        ;
run;

/*使用 HASH 筛选新员工的销售额*/
Data   sales_newly;
        Length   id   $8   epl_ym   8.;
        If   _n_=1   then   do;
          Declare   hash   newly(dataset:'id_newly');
                newly.definekey('id');
                newly.definedata('epl_ym');
                newly.definedone();
        end;
        set   sales_all;
/*数据集 sales_all 中的 id 值能够在 HASH 对象 newly 中检索到*/
        rc=newly.find(key:id);
        if   rc=0;
run;
```

【程序解读】

1）使用数据集 id_newly 定义 HASH 对象，id 为主键，信息变量为 epl_ym。

2）newly.find(key:id)是调用 FIND 方法检索数据集 sales_all 中变量 id 的值是否出现在 HASH 对象主键中。如果括号中的内容为空，SAS 默认自动匹配 HASH 对象的主键和数据集中的同名变量来检索。

3）IF 语句是将检索到的 id 也就是两者的交集输出到结果中。

结果显示如表 12-11 所示。

表 12-11　数据集 sales-newly 信息显示

id	epl_ym	grade	amt	rc
1101	201201	C	256	0
1102	201201	C	128	0
1123	201203	D	96	0
1105	201202	C	196	0
1104	201202	D	89	0
1115	201202	C	221	0

【例 12.13】 使用 HASH 对象来拼接数据。接上例，根据销售级别对应佣金比例参数表，计算新员工销售额并计算佣金。

```
/*销售等级对应佣金比例*/
data   grade;
       input   grade   $   rate;
       cards;
       A 0.20
       B 0.18
       C 0.15
       D 0.10
       E 0.05
       ;
run;

/*计算佣金*/
Data   csm_amt;
        if _n_=0   then   do;
        set   id_ newly   grade;
        end;
       else   if   _n_=1   then   do;
        declare   hash   newly(dataset:'id_newly');
             newly.definekey('id');
             newly.definedata('epl_ym');
             newly.definedone();
        declare   hash   grd(dataset:'grade');
             grd.definekey('grade');
             grd.definedata('rate');
             grd.definedone();
```

```
        end;
/*      call    missing (of _all_);*/
        set    sales_all;
        rc1=newly.find(key:id);
        rc2=grd.find(key:grade);
        if rc1=0 then csm_amt=amt*rate;
run;
```

【程序解读】

1）使用数据集 id_newly 定义 HASH 对象 newly。

2）使用数据集 grade 定义 HASH 对象 grd。

3）调用 FIND 方法拼接 id 对应的 emp_ym 和 grade 对应的 rate。

4）如果是新员工，计算佣金。

结果显示如表 12-12 所示。

表 12-12　数据集 csm-amt 信息显示

id	epl_ym	grade	rate	amt	csm_amt
1001		A	0.2	561	
1101	201201	C	0.15	256	38.4
1002	201201	B	0.18	421	
1003	201201	A	0.2	691	
1005	201201	A	0.2	555	
1004	201201	B	0.18	398	
1015	201201	A	0.2	402	
1102	201201	C	0.15	128	19.2
1123	201203	D	0.1	96	9.6
1105	201202	C	0.15	196	29.4
1104	201202	D	0.1	89	8.9
1086	201202	B	0.18	632	
1093	201202	A	0.2	701	
1115	201202	C	0.15	221	33.15

通过结果可以看出每位新员工都计算了佣金金额，而非新员工此字段为缺失。但是发现除第一条观测外非新员工的入职月份 epl_ym 字段也有值，原因何在呢？本章开头了解过 PDV，知道 PDV 中的变量并不会自动清除数值，当新的观测读入时，如果 PDV 中变量的数值没有被更新则此数值会被保留并加入到读入观测中去。在上例中，从数据集 sales_all 中读取第二条观测时，id 等于 1101 可以在 HASH 对象中检索到，同时 emp_ly 的数值被读入到 PDV 中。当读入第三条观测时，虽然 id 等于 1002 在 HASH 对象中没有找到，但是由于 emp_ly 的数值并没有清除，此数值被保留下来并且加入到了第三条观测中。要解决这个问题，需要调用 MISSING 函数，在下一次读入观测前将变量数值清除。将程序中的语句 call missing (of _all_) 反注释后运行结果如表 12-13 所示。

表 12-13　加 MISSING 函数后数据集 csm-amt 信息显示

id	epl_ym	grade	rate	amt	csm_amt
1001		A	0.2	561	
1101	201201	C	0.15	256	38.4
1002		B	0.18	421	
1003		A	0.2	691	
1005		A	0.2	555	
1004		B	0.18	398	
1015		A	0.2	402	
1102	201201	C	0.15	128	19.2
1123	201203	D	0.1	96	9.6
1105	201202	C	0.15	196	29.4
1104	201202	D	0.1	89	8.9
1086		B	0.18	632	
1093		A	0.2	701	
1115	201202	C	0.15	221	33.15

本例中在指定变量长度时，利用 SAS 先编译后执行的原理通过一次性指定变量长度。如果定义 HASH 对象的数据集中信息变量个数比较多，则可以在调用 DEFINEDATA 时使用“ALL”选项。

【例 12.14】 调用 DEFINEDATA 时使用“ALL”选项。

```
/*小组信息*/
data  team_info;
      input  team  $  open_ym  grade  employees;
      cards;
      A 201201 12 6
      B 201202 11 9
      C 201205 9 12
      D 201206 7 20
      E 201205 8 9
      F 201206 10 23
      ;
run;
data  employ_new;
      input  id  $  team  $;
      cards;
      2001 C
      2036 D
      ;
run;
data  newly_info;
      if  _n_=0  then  do;
```

```
se t team_info;
end;
        else    if    _n_=1    then    do;
          declare    hash    t(dataset:'team_info');
                t.definekey('team');
                t.definedata(all:'yes');
                t.definedone();
        end;
        call    missing (of _all_);
        set    employ_new;
        rc=t.find();
run;
```

HASH 对象的应用和自定义 FORMAT 格式的应用有很多相似之处，在数据查询时省去了很多数据集之间的拼接，减少了数据处理过程中 SORT 和 MERGE 的使用，提高了数据处理效率。但是两者有所区别，HASH 对象的效率更高一些，但 HASH 对象每次都需要在 DATA 步过程中定义，并会消耗内存，所以定义 HASH 对象的数据集应尽量控制观测数和变量数量。而自定义 FORMAT 格式虽然内容丰富、应用灵活，可以一次定义反复调用，但 FORMAT 格式的存储和调用同样需要消耗存储空间，所以自定义 FORMAT 格式时也需要考虑资源消耗的问题。

参 考 文 献

[1] 杨池然．SAS9.2 从入门到精通[M]. 北京：电子工业出版社，2011．
[2] 董大均．SAS 统计分析应用[M]. 北京：电子工业出版社，2008．
[3] Jiawei Han, Micheline Kamber. 数据挖掘概念与技术[M]. 范明、孟小峰，等译. 北京：机械工业出版社，2001．
[4] 朱世武．SAS 编程技术教程[M]. 北京：清华大学出版社，2007．
[5] W. H. Inmon.数据仓库[M]. 王志海，等译. 北京：机械工业出版社，2000．
[6] 陈希孺.概率论与数理统计[M]. 合肥：中国科学技术大学出版社，2009．

在线互动交流平台
官方微博：http://weibo.com/cmpjsj
豆 瓣 网：http://site.douban.com/139085/
读者信箱：cmp_itbook@163.com

SAS开发
经典案例解析

作者简介



杨池然：现就职于某商业银行，具备多年银行业SAS与数据仓库项目经验，曾参与大量SAS大数据量数据处理、SAS统计分析、数据挖掘、SAS与Oracle数据仓库交互应用项目的开发与实施。研究领域：SAS编程开发、SAS与统计分析、SAS与各类数据处理、数据挖掘、SAS与数据仓库交互应用开发、SQL结构化查询语言开发等。曾出版专著：《SAS9.2从入门到精通》。

地址：北京市百万庄大街22号
邮政编码：100037
电话服务
社服务中心：010-88361066
销售一部：010-68326294
销售二部：010-88379649
读者购书热线：010-88379203
网络服务
教材网：http://www.cmpedu.com
机工官网：http://www.cmpbook.com
机工官博：http://weibo.com/cmp1952
封面无防伪标均为盗版

上架指导 计算机/数据分析
ISBN 978-7-111-41100-0
策划编辑◎丁 诚 / 封面设计◎





ISBN 978-7-111-41100-0
定价：59.00元